Online handwritten Kannada word recognizer with unrestricted vocabulary

Rituraj Kunwar, Shashikiran K, A. G. Ramakrishnan
MILE Lab, Department of Electrical Engineering, IISc, Bangalore, India.
{*kunwar.rituraj, shashi.reach, agrkrish*} *@gmail.com*

Abstract

In this paper, we propose a novel heuristic approach to segment recognizable symbols from online Kannada word data and perform recognition of entire word. Two different estimates of 1st derivative are extracted from the preprocessed stroke groups and used as features for classification. Estimate 2 proved better with 88% accuracy, which is 3% more than that achieved with estimate 1. Classification is performed by the Statistical DTW (SDTW) classifier which uses X, Y co-ordinates and their 1st derivatives as features. Classifier is trained with 40 writer data making it writer independent. 295 classes are handled covering Kannada aksharas, with Kannada numerals, Indo-Arabic numerals, punctuations and other special symbols like $, # etc. Akshara level classification accuracy is 88% and a lower accuracy of 80% *at word level, which shows the scope for further improvement in segmentation algorithm.*

1. Introduction

Rapid development in technology has made handheld devices very popular and in coming days with increase in demand, it will be affordable too. This will lead to handwriting recognition being an alternative to the keyboard as an input device, along with, may be, speech recognition. Data entry using pen is a natural form of interface. Various techniques have been explored for handwriting recognition and efficient practical applications exist for English. However, there is no such system for Indian languages so far. 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, or approaches based on limited vocabulary lexicon based recognizers using HMM. While the above approaches have their own applicability, when it comes to unlimited vocabulary recognition involving proper names, addresses, etc. the above approaches simply cannot be used at all.

In this paper, we aim to develop a system that recognizes at the word level. No constraint has been imposed on the type of word to be recognized. This is made possible by developing a robust segmentation algorithm which segments the word level data into recognizable symbols (stroke groups) and later performing the recognition at the stroke group level. At the end, recognized output is combined to give the Unicode of the recognized word. The complete set of Kannada aksharas, including the Kannada and Indo-Arabic numerals, besides other symbols are handled

Word recognition can be performed in various ways. One of the popular approaches is to obtain a model of each word and the whole word is recognized as an entity. The drawback of this approach is that it cannot handle the proper nouns since capturing all possible nouns is near to impossible. Another bottleneck of this approach is the computation time due to the enormously large number of training classes. An alternate approach to perform recognition at word level is by enforcing the writer to write the recognizable symbols far apart so that segmentation of symbols remains no more a challenge. But it is obvious that this approach basically does not handle word as this approach will fail if we move a step forward that is to carry out recognition at the sentence level. Thus, it will not be able to segment 2 different words in a sentence.

In the proposed method, all the issues mentioned above have been addressed by using a heuristic segmentation algorithm to segment recognizable symbols (group of strokes) from word level input. This algorithm exploits the basic characteristics of Kannada script. The proposed algorithm looks into the relative positions of the stroke groups and depending on the values of thresholds, they are segmented. The value of these thresholds are set empirically and this characteristic of the algorithm makes it script dependent. Further, each segmented symbol is fed to the recognizer, which gives a class label to it. Finally, all the class labels of the segmented symbols are combined using the rules of Kannada script grammer to generate the Unicode string

of the recognized word.

2. Kannada OHR Survey

2.1. Kannada Script:

Kannada is the official language of the South Indian state of Karnataka. It has its own script derived from Bramhi script. Kannada script has a base set of 52 characters, comprising 16 vowels and 36 consonants. Further there are distinct symbols that modify the base consonants, called consonant and vowel modifiers. The number of these modifiers is the same as that of the base characters. The characters called aksharas are formed by graphically combining the symbols corresponding to consonants, consonant modifiers (optional) and vowel modifiers using well defined rules of combination. Therefore, the number of theoretically possible combinations of Kannada characters is 16 vowels, 36*16=576 consonant-vowel combinations and 36*36*16=20736 consonant-consonant-vowel combinations.

While designing a character recognition system, if we consider each akshara as a separate class, the number of classes becomes prohibitively high. However in Kannada, all consonant modifiers are written separate from the base character and at least some part of some stroke of a consonant modifier will lie below the base character as shown in Fig.1. So we consider the consonant modifiers as separate classes. This reduces 36*36*16 C-C-V combinations (or possible classes) to 36*16 C-V combinations of the base character and additional 36 classes of the consonant modifiers. Similarly,

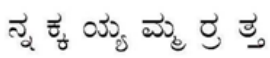

Figure 1. Consonant modifiers lie below the base character

some of the vowel modifiers are also written separately from the base character as shown in Fig. 2 . Thus, considering these as separate classes, we reduced the total number of classes further. In all, we reduced the total number of classes to 295 including Indo-Arabic numerals, Kannada numerals and punctuation marks. By recognizing these symbols, we cover the whole of the Kannada character set. One of the applications we have in mind for our recognizer is that of a form-filling application, which necessarily has names, numbers, punctuation marks and special symbols. Hence, we include all of the above as pattern classes.

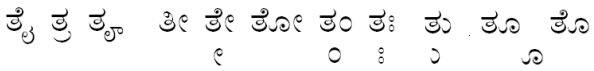

Figure 2. Examples of some of the vowel modifiers written separately from the base character, which could therefore be segmented from the character complex (akshara) and recognised separately.

2.2. Status of Kannada OHR:

Work on Kannada OCR and OHR have been few and far between. One of the few works reported in Kannada character OCR was by P.S. Sastry et. al. [1] which was font and size independent with reported performance between 80 to 86%. Another work done on OCR of Kannada character was by Vijay Kumar et. al. [6] using neural network with accuracy reported to be 95%. Similar work on Kannada offline numeral handwritten recognition was conducted by S.V. Rajashekararadahya et. al. [9] with reported performance of 95%. One of the works on Kannada character OHR is by S.R. Kunte et. al. [7] using wavelet features and neural network as classifier, reporting an accuracy of 95%. However in none of the above cases, details of the dataset used are given, rendering the different works incomparable.

To the best of our knowledge, this is the first work in Kannada OHR, where the recognition is done at the word level, covering the complete character set of Kannada: base characters, vowel modifiers, ottus and numerals (both Kannada and Indo-arabic). It is difficult to infer few details from the previous works in Kannada OHR as to whether they are writer dependent or independent and how efficient they are in terms of time.

In the present work, we have addressed these issues. We developed a robust segmentation algorithm which conforms to the strategy used to reduce the number of classes. Our Kannada OHR engine has been trained with data from 40 users thus making it writer independent. In spite of covering all the characters of Kannada, symbols, etc. we managed to have the number of classes manageable by exploiting the nature of the script as discussed in sub-section 2.1, above.

3. Classification Experiments

The building blocks in Online Handwriting Recognition is shown in Figure 3. The data collected or captured using the Tablet PC is considered as the raw data. This raw word level data is given as input to the segmentation algorithm. Each word is now segmented into

recognizable stroke groups. The raw data can be noisy due to erratic hand movements while writing. Hence the obtained stroke groups need to be smoothed. This is done by preprocessing, normalizing and resampling the data. Resampling is based on equi-arc length. The preprocessed data is used to extract features of each stroke group, which are then used for the classification of that particular stroke group. Each segmented stroke group is recognized similarly. The Unicodes of all the recognized symbols are combined to give the Unicode of the word.

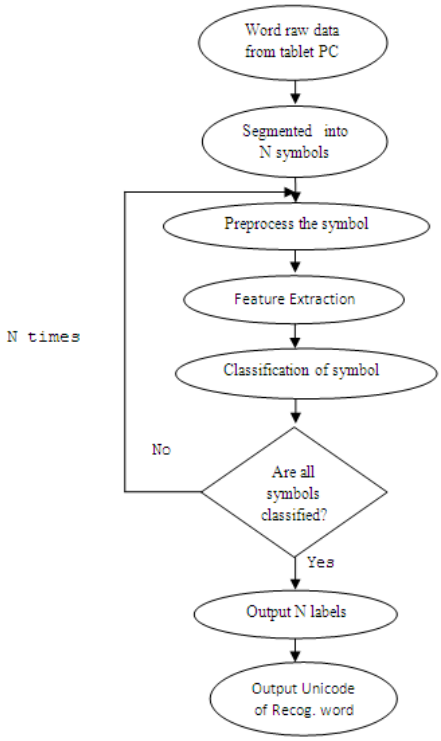

Figure 3. Schematic of Kannada OHR experiments.

3.1. Segmentation

Challenges in segmentation: Enormous amount of variability in the handwriting style is observed. This necessitates a robust segmentation method. The challenges involved in segmentation are as follows:

Spatial and temporal information is used in segmentation as well as in recognition methods. This information gets affected due to the following issues, thereby resulting in a drop in the recognition accuracy.

- **Correction**: In the real time data, sometimes a writer makes correction in one or more of the previously written aksharas. An example of this case is shown in Fig 4.1.

- **Unusual stroke order**: It is observed that people who do not write the script regularly tend to change the usual order of strokes to suit their convenience. An example of this case is shown in Fig 4.2

- **Overwriting:** Imperfections of the data collection devices results in improper feedback making the writer overwrite one or more strokes, which lead to misclassification. An example of this case is shown in Fig 4.3.

- **Delayed strokes:** Kannada language contains strokes like paadam and dot, which a few users end up writing at the last. This is similar to the common practice of marking the t's or placing the dot of i's at the end in English language. This leads to alteration in the temporal information. An example of this case is shown in Fig 4.4.

- **Strokes in reverse direction:** Some writers change the direction of writing a stroke for ease of writing or because they have been out of practice. This creates confusion in segmenting correct stroke groups. An example of this case is shown in Fig 4.5.

- **Merged strokes:** Apart from this, users also combine two or more strokes which are expected to be written separately or vice-versa as shown in Fig 4.6.

An observation of the stroke order and the position of the strokes of an akshara in Kannada, gives us an idea of how segmentation strategy can be used for grouping strokes into different recognizable symbols.

Base Characters: We observe that in general, majority of the symbols are multi-stroke as shown in Fig: 5.1. The strokes of a multi-stroke symbol occur with a dominant horizontal overlap and the stroke order is usually from bottom to top.

Conjuncts ("ottu"s in Kannada): These are vowel and consonant modifiers that occur below the base symbol. Parts of most Ottus occur below the character; there is generally an overlap in the horizontal axis and the ottu starts below the base character i.e. first point of the stroke forming the Ottu is below the base character/symbol it is associated with as shown in Fig.5.2.

Pulli/Paadam in Kannada: This is a small vertical line at the bottom of a consonant which changes it from an unaspirated (alpaprana) to an aspirated (mahaprana) consonant. as shown in Fig.5.3.

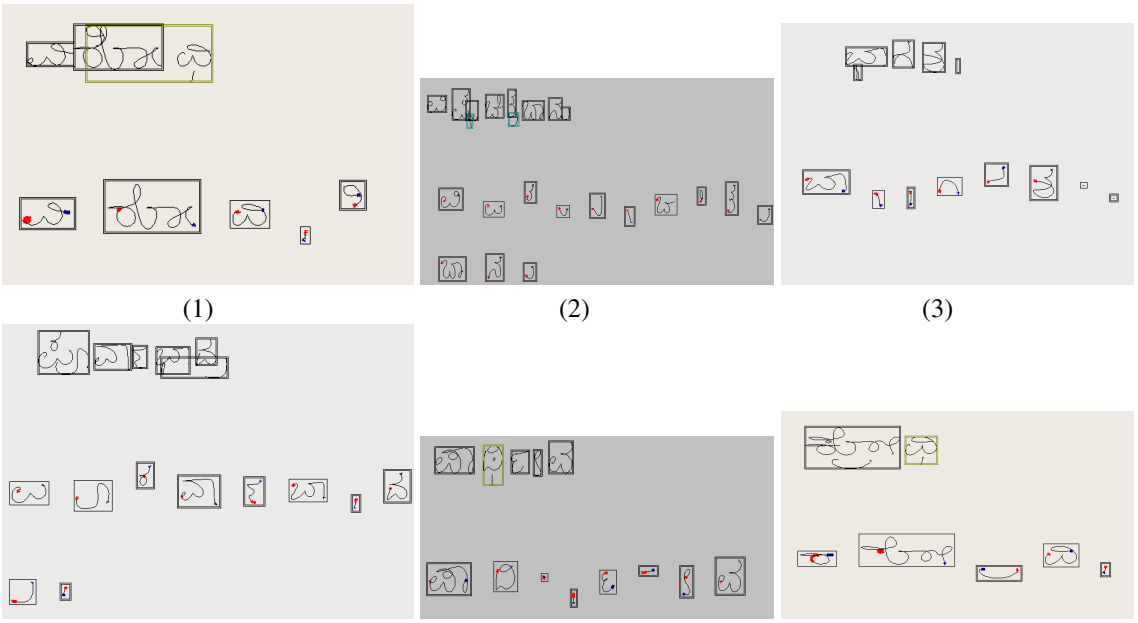

Figure 4. (1) Correction of strokes (2) Change in stroke order (3) Overwriting of strokes (4) Delayed strokes (5) Strokes written in reverse direction (6) Merged strokes.

Dots in Kannada script: In Kannada, a dot can occur meaningfully as part of two aksharas, namely Tha and tha. As shown in Fig.5.4, the first symbol 'ra' becomes 'Tha' by the addition of the dot inside the loop. Such dots are identified by size and position and removed and a corresponding flag is set. Recognized symbol is assigned a label based on the status of this flag.

3.2. Feature extraction

In this module some high level structural features are extracted from the preprocessed stroke group. The estimates of derivative features extracted are defined as.

Estimate 1 of First Derivative: In this method, the derivative at the current point is estimated using the formula given below:

$$X'_1(j)=\frac{\sum_1^2 i\,[x(j+i)-x(j-i)]}{2\sum_1^2 i^2}, Y'_1(j)=\frac{\sum_1^2 i\,[y(j+i)-y(j-i)]}{2\sum_1^2 i^2} \quad (1)$$

Since the above formula cannot be estimated for the 1^{st}, 2^{nd}, last and 2^{nd} last points, their values are calculated after extending the length of the sequence.

where $1 \leq j \leq$ L where L = number of points in the pattern. X'(1)=X'(2)=X'(3); Y'(1)=Y'(2)=Y'(3); X'(L)=X'(L-1)=X'(L-2); Y'(L)=Y'(L-1)=Y'(L-2)

Estimate 2 of first Derivative Estimate 2: Another estimate of derivatives of *x* and *y* at each point was suggested as a feature in [5]. The estimated derivative of *x* and *y* at the point *j* in a pattern could be calculated using the formulae given below:

$$X'_2(j)=\frac{[x(j)-x(j-i)]+\frac{[x(j+1)-x(j-i)]}{2}}{2}, \quad (2)$$

$$Y'_2(j)=\frac{[y(j)-y(j-i)]+\frac{[y(j+1)-y(j-i)]}{2}}{2}, \quad (3)$$

Since the above formula cannot be estimated for the first and last points, their values are assumed to be the same as those of second and penultimate points, respectively:

X'(1)=X'(2); Y'(1)=Y'(2); X'(l)=X' (L-1); Y'(l)=Y'(L-1) where $1 \leq j \leq$ L where L = number of points in the pattern.

3.3. Statistical Dynamic Time Warping (SDTW)

SDTW: In SDTW, a reference character is represented by a sequence Q = $(Q^1, Q^2, Q^3, ...Q^{lq})$ of statistical quantities (states) [2], as shown in Fig 6. These statistical quantities include

1) Discrete probabilities say α^j : Ω[0,1] for statistical modeling of transitions $\Delta\phi\epsilon\Omega$ reaching the sequence's state *j*. i.e. state transition probabilities are defined by

In the special case of $n = j = 1$,

$$\alpha_j(\Delta\phi)=P(\Delta\phi=\phi(n)-\phi(n-1)|\phi_R(n)=j) \quad , \quad \Delta\phi\epsilon\Omega \quad (4)$$

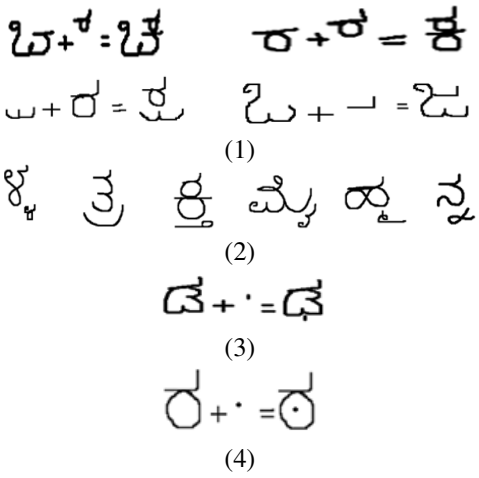

Figure 5. (1)Base Character (2)Conjuncts (ottus) (3) Pulli/Paadam (4) Dots.

$$\alpha_1(1,1)=P(\Delta\phi(1)=\phi(1)-\phi(0)|\phi_R(1)=1)=1 \quad (5)$$

2) A continuous probability density function β_j : $R^d \longrightarrow R$ that models the feature distribution at sequence's state *j*. In our work we modeled β_j by a unimodal, multivariate Gaussian distribution i.e.

$$\beta_j(x) = P(x|\phi_R(n) = j)=N_{\mu_j,\Sigma_j}(x)=\left((2\pi)^d|\Sigma_j|\exp\left((x-\mu_j)^T\Sigma_j^{-1}(x-\mu_j)\right)\right)^{\frac{-1}{2}} \quad (6)$$

In some situations, the probability $\alpha_j'(\Delta\phi)$ that a transition $\Delta\phi$ emerges from the sequence's state *j* is also required and is related to $\alpha_j(\Delta\phi)$ as

$$\alpha_j(\Delta\phi) = \alpha'_{j-\Delta\phi_R}(\Delta\phi)\ and\ \sum_{\Delta\phi}\alpha'_j(\Delta\phi) = 1 \quad (7)$$

While testing, the SDTW distance of test pattern to the reference model of each class is computed and the test pattern is assigned the label of the class giving minimum SDTW distance. Here, the definition of SDTW distance is somewhat different from that of DTW and is given by

$$SD_\phi*(T,Q) = min_\phi SD_\phi(T,Q) \quad (8)$$

where SD_ϕ(T,Q) is statistical warping distance between pattern T and model Q, with Φ as alignment path and is given by

$$SD_\phi(T,Q) = \sum_{i=1}^{N}(d(t_{\phi_t(i)},Q_{\phi_q(i)}) - log_e\alpha_{\phi_q(i)}(\Delta\phi(i))) \quad (9)$$

where

$$d(t_i,Q_j) = 0.5((t_i-\mu_j)^T\sum_j^{-1}(t_i-\mu_j)+dlog_e(2\pi)+log_e|\sum_j|) \quad (10)$$

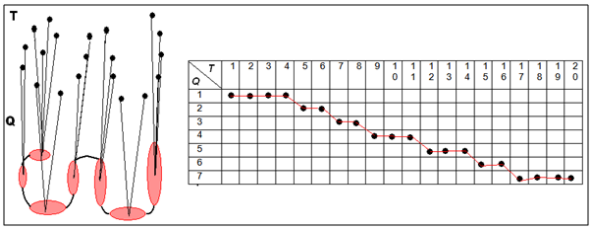

Figure 6. Transitions between states in SDTW

Fig.6 shows how the matching takes place between the reference model and the test pattern. The matrix in Fig.6 shows the SDTW path (path of best SDTW matching).

It can be observed that the SDTW distance is the negative log state optimized likelihood of pattern T generated by the model Q, with optimal state sequence Φ^* given by the Viterbi algorithm. So, models in SDTW framework are similar to HMMs [8], [3] of particular type with state prior probabilities $\pi = (1,0,0,....0)^T$ and are of left to right models with step size of at most 1 and with null transitions (transitions that allow change in state without observation change i.e. transitions (0,1) in Ω). So the models in SDTW framework can be trained by algorithms used for training HMMs. In our work, we used segmental K-means algorithm [3] for training SDTW model parameters.

4. Data collection for our experiments

MILE lab Kannada database for conducting the experiment was built by collecting data from 69 different writers. Thus the recognition engine could be trained with different styles of handwriting to make the OHR engine writer independent. Each writer wrote all the symbols corresponding to the 295 classes. Writers for data collection were meticulously chosen: only ones who regularly use Kannada script. Justification of this constraint is that we need to capture the regular style of Kannada handwriting, which includes local temporal information.

5. Results

The accuracy of the classifier was evaluated on the Kannada Database of Medical Intelligence and Language Engineering (MILE) Lab, IISc. The results are

Table 1. Writer independent symbol level recognition performance of SDTW with different estimates of first derivative as features. (No. of classes: 295; No. of training and testing samples/class: 40 and 29, respectively.)

Features used	Symbol Accuracy (%)	Recognition time/sample (sec)
Preprocessed X,Y co−ordinates and 1st derivatives of X,Y co-ordinates (Estimate 1)	85.2	0.5
Preprocessed X,Y co−ordinates and 1st derivatives of X,Y co-ordinates (Estimate 2)	87.9	0.5

Table 2. Writer independent word level recognition performance on randomly collected 100 words using SDTW as classifier with estimate 2 of first derivatives as features.

Classifier	Features used	Unicode level Accuracy (%)
Statistical DTW	Preprocessed X,Y co−ordinates and 1st derivatives of X,Y co-ordinates (Estimate 2)	80

shown in Table 1. Kannada dataset has 295 classes where each class represents a group of strokes. Each class has been trained and tested with 40 and 29 samples files, respectively.

As seen in Table 1, estimate 2 of first derivative beats estimate 1 by approximately 3% with same performance in terms of time. Using the segmentation algorithm, recognition at word level has been evaluated using the estimate 2 as feature and SDTW as classifier as shown in Table 2. Word level accuracy is 80%, which has gone down as compared to symbol level accuracy. This is because few of the segmentation challenges listed in section 3.1 have not yet been dealt with completely. It is evident from our results shown in Table 1 that our above approach can be applied in real time applications.

6. Conclusions

We have demonstrated the effectiveness of segmentation in word recognition. To our best knowledge, this is the first work that deals with Kannada word level recognition handling all combinations of consonants and vowels, punctuations, Kannada and Indo-Arabic numerals appearing in a word. Currently, we are working to handle all the segmentation challenges listed above in section 3.1 and improve the accuracy at the word level.

References

[1] T. V. Ashwin and P. S. Sastry. A font and size-independent ocr system for printed kannada documents using support vector machines. *Sadhana*, 27(1):35–58, 2002.

[2] C. Bahlmann and H. Burkhardt. The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic time warping. *IEEE Trans. PAMI*, 26(3):299–310, March 2004.

[3] R. Dugad and U. B. Desai. A tutorial on hidden markov models. *SPANN*, May 1996.

[4] N. Joshi, G. Sita, A. G. Ramakrishnan, and S. Madhvanath. Comparison of elastic matching algorithms for online tamil handwritten character recognition. *Proc intl. workshop frontiers handwriting recog.*, pages 444–449, October 2004.

[5] E. Keogh and M. Pazzani. Derivative dynamic time warping. *First SIAM Intl. Conf. Data Mining*, 2001.

[6] B. V. Kumar and A. G. Ramakrishnan. Machine recognition of printed kannada text. *Proc. Intl. workshop doc. anal. systems*, pages 37–48, 2002.

[7] S. R. Kunte and S. Samuel. Wavelet features based online recognition of handwritten kannada characters. *Jl Visualization Society of Japan*, pages 417–420, 2000.

[8] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–286, February 1989.

[9] S. V. Rajashekararadahya and P. V. Ranjan. Handwritten numeral recognition of three popular south indian scripts. *Proc intl. conf. information processing*, pages 162–167, 2008.

[10] A. K. J. Scott D Connell. Template based online character recognition. *Pattern Recognition*, 34:1–14, 2001.