Recognition of Eyes-free Handwriting Input for Pen and Touch Interfaces

Bharath A. and Sriganesh Madhvanath
Hewlett-Packard Labs India
Bangalore, India
{bharath.a, srig}@hp.com

Abstract

Continuous handwriting input using the finger on small devices such as PDAs and mobile phones is difficult if not impossible. Also, handwriting input has always required significant attention from the user during the process of writing. In our previous work, we proposed the concept of 'eyes-free writing', a novel solution that addresses both the problems. In this paper, we describe the challenges from the perspective of recognizing eyes-free handwriting input and explain the architecture of our recognition system in detail. Recognition is performed using word-level Hidden Markov Models (HMM), and empirical evaluation of the recognition accuracy using English and Tamil handwriting datasets has shown promising results.

Keywords: Text input, pen and touch interfaces, handwriting recognition.

1. Introduction

The naturalness or ease-of-use offered by handwriting-based text input methods has been the motivation for researchers for several years to explore pen-based human-computer interaction. Existing handwriting-based interfaces can broadly be classified into character-by-character input and continuous (e.g., word) input. Character-by-character entry methods typically either make use of the time lapse after writing or await explicit indication by the user to trigger recognition. The Unistroke approach [4] uses the end of stroke to signify the end of a character and provides "heads-up" text input, but require the user to learn special symbols. Word input methods allow the entire word to be written at once before invoking recognition. As a result, continuous handwriting input methods for text entry are more efficient than their character-entry counterparts. Apart from the challenges in recognizing continuous input, in recent times as the sizes of the devices tend to shrink, continuous handwriting using a finger on a small area such as a PDA or a notebook touchpad becomes difficult.

A key aspect in the case of handwritten input is user attention. Almost in all continuous input methods such as the Microsoft's TabletPC handwriting input panel, the user writes the way he/she would normally do on paper. The characters that make up the word are written one after another, with similar sizes and aligned horizontally with respect to an imaginary or displayed baseline. This classical way of writing demands significant user attention and burdens the user with constant context switching between the input panel and the application in focus. The need for user attention becomes even more important in the absence of visual feedback from the device, as in the case of a graphics tablet.

The remainder of the paper is organized as follows. In the next section, we describe the solution proposed in our previous work [1] which addresses the two main issues of continuous handwriting input using the finger on a small device and the need for significant user attention during the process of writing. In the sections that follow we focus on the different components of a modified Hidden Markov Model based word recognition system developed to recognize eyes-free handwritten input. Section 3 describes the preprocessing steps. The features extracted from the ink are explained in Section 4. Word modeling using HMMs is illustrated in Section 5. Sections 6 and 7 describe the datasets used and the experimental evaluation carried out. Some conclusions and future directions are stated in the final section.

2. Eyes-free Handwriting Input

Our solution to the problems described above is "attention-free writing" or "eyes-free writing". This solution is inspired by the question: what if the user does not look at the writing? In other words, what are the side-effects of such handwritten input? Analogous to touch-typing where the typist does not look at the keyboard while entering text, in the case of "eyes-free writing" the writer does not have to look at the writing surface. Some of the issues one may envisage resulting from such inputs are:

a. The relative position between the strokes forming a character and between characters

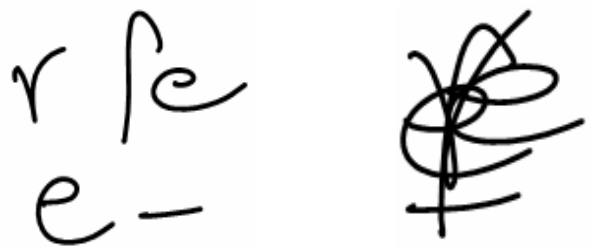
becomes highly unreliable. As a result, the shape of a multi-stroke character on the whole becomes worse and the spatial alignment between characters is lost.

b. The ascenders and descenders, considered to be vital information in some approaches for word recognition, disappear.

The eyes-free writing model may also be logically extended to the small writing surface scenario (e.g. notebook touchpad, mobile phones etc.) wherein the user is allowed to overwrite on the same surface without attempting to lay characters out in the conventional left-to-right order. Such handwriting input is possible only because of the intrinsic advantage with online ink capture. The core idea of eyes-free writing is to use the shapes and sequence of strokes, and ignore the unreliable sizes and relative positions. Fig. 1 shows some examples of eyes-free writing for large and small writing surface scenarios.

A related effort we came across recently is "overlaid handwriting" published in [8]. That effort also addresses the problem of small real-estate by allowing the users to write one character over another to form the word. However, it assumes that visual feedback is available, and the relative position between strokes forming a character is maintained by the user. In contrast, our solution does not require that the ink be rendered back to the user, or that relative position between strokes be maintained. In addition to enabling "eyes-free" text input, this also makes our solution suitable for devices without graphical displays.

The following sections describe the different stages of our recognition system and how we cope with the challenges posed by eyes-free writing.



(a) (b)

Figure 1. Eyes-free writing of the word "free" on a (a) large surface (b) small device

3. Preprocessing for Eyes-free Handwriting Input

Typically in any handwriting recognition system, preprocessing is carried out to eliminate noise and any undesirable variations in the input. However, unlike the preprocessing techniques presented in the literature, in order to cope with the issues described in the previous section, we carry out *stroke-level preprocessing*. To deal with the unreliable nature of position information, irrespective of the position of the stroke on the writing surface, we translate each stroke such that the (x-min, y-min) point of its bounding box is moved to the origin. In order to cope with the unreliable sizes, each stroke is normalized by fixing the height or width of the bounding box, whichever is larger, to a constant value and then altering the length of the other dimension taking into account the original aspect ratio of the stroke. Once the handwritten word is preprocessed at the stroke level, each stroke in the temporal order appears to be written at the same location and with similar size as shown in Fig. 2.

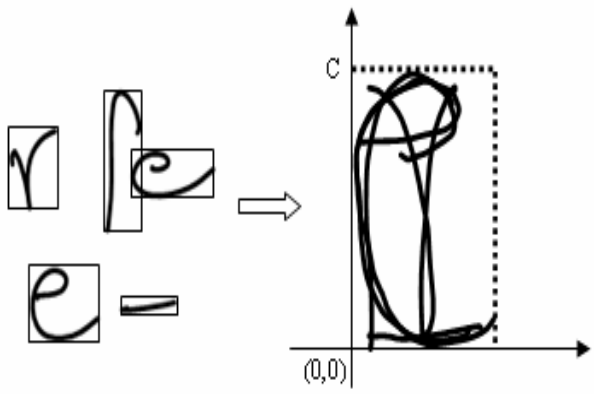


Figure 2. Stroke-level preprocessing

4. Feature Extraction

Once the strokes in the word are preprocessed at the stroke level, features are extracted at each point. Time-domain features described in [6] are shown to be speed-invariant and hence even for writer-independent recognition no resampling is required in the preprocessing step. The features at each point in the trajectory include size normalized values of x and y, normalized first and second derivatives of x and y, and curvature. The normalized first derivatives are computed using the equations shown below. The normalized second derivatives are obtained by replacing x and y with the first derivatives in these equations.

$$x'_t = \frac{\sum_{i=1}^{2} i.(x_{t+i} - x_{t-i})}{2.\sum_{i=1}^{2} i^2} \qquad y'_t = \frac{\sum_{i=1}^{2} i.(y_{t+i} - y_{t-i})}{2.\sum_{i=1}^{2} i^2}$$

$$x'_{N_t} = \frac{x'_t}{\sqrt{x'^2_t + y'^2_t}} \qquad y'_{N_t} = \frac{y'_t}{\sqrt{x'^2_t + y'^2_t}}$$

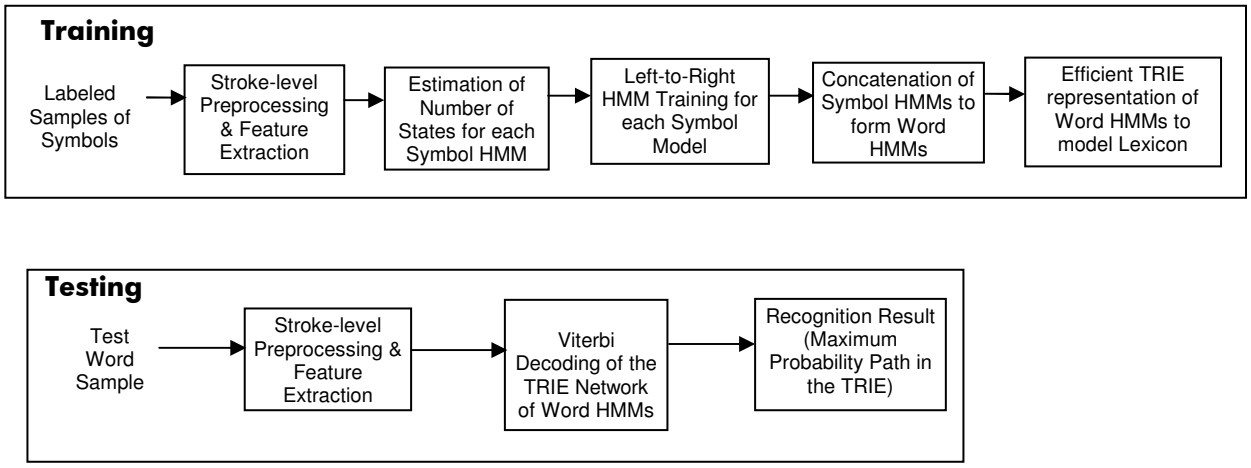


Figure 3. HMM-based architecture for recognizing eyes-free writing [1]

5. HMM-based Recognition System

The HMM framework briefly described in [1] for recognizing English overwriting, is described here in detail and further validated for Tamil, an Indic script. The different steps involved in training and testing of our Hidden Markov Model based word recognition system are shown in Fig. 3.

5.1. Symbol Definition

In the recognition of eyes-free writing for English, lowercase letters (a-z) are defined as the symbols. On the other hand, Tamil script belongs to the family of syllabic alphabets [3] and consists of symbols for vowels and consonants. In Tamil, the implicit vowel sound of each consonant can be modified by adding another vowel sound in the form of a diacritical mark, also known as *matra*. The symbol set defined in [2], shown in Fig. 4, is reused here to form the basic set of symbol classes in Tamil. Both in the case of English and Tamil, symbols are the fundamental units of recognition which are then extended to represent a syllabic unit (consonant vowel combination) in the case of Tamil, and then a word. However, representing a Unicode word in Tamil as a sequence of these symbol IDs implicitly imposes a writing order. Based on our knowledge about the script and manual investigation of the collected handwriting samples, we find that most people write the consonant first followed by the matra barring few exceptions. Assuming this as a constraint, any Unicode string in Tamil can be converted into a unique sequence of symbol IDs.

5.2. Symbol Modeling

Hidden Markov Models are used for handwriting recognition for two main reasons. Firstly, these are stochastic models and hence can cope with noise and variations in the input. Secondly, HMMs solve the problem of segmentation implicitly.

Figure 4. Tamil symbol set defined for word recognition [2]

In the case of Tamil, handwritten word samples collected from different writers are cleaned and labeled at the symbol level. From the labeled word samples the ink samples corresponding to each symbol are extracted. As a result, each symbol typically has a few hundred samples which form the training data for that symbol. In the case of English, isolated lowercase character samples from the IRONOFF [9] dataset form the train set. These samples are then used to train a left-to-right HMM with no state skipping (Fig. 5) to model the symbol. The well-known Baum-Welch algorithm [7] is employed for training. The number of states for the symbol HMM is determined based on the average trajectory length of the symbol in the training samples. The trajectory length is expected to indicate the complexity of the shape as higher the length more complex the shape is. Therefore complex shapes are modeled using a large number of states. The state probability density function is assumed to be a mixture of Gaussians and the number of components was determined empirically to be *15*.

The reader may note that in the symbol set for Tamil (Fig. 4) symbol 72 (vowel muting diacritic) and symbol 83 (period) are of identical shape and differ only in their absolute positions in the word context. But since our preprocessing step discards the position information, the samples of 72 and 83 are used to train a single HMM and the model is shared between the two symbols. However, sometimes the loss of position information may also result in ambiguity. For instance, taking an example from English, it becomes impossible to distinguish a two-stroke 'b' written as "l" followed by ")" from a two-stroke "p" written in the same way. In such cases additional contextual information either in the form of a lexicon or a language model is needed to resolve the ambiguity.

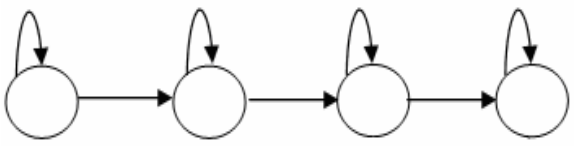


Figure 5. Topology of a symbol HMM

5.3. Word and Lexicon Modeling

Once the symbol models are formed as described above, each word entry in the lexicon is again modeled as an HMM. The word model corresponding to a sequence of symbol IDs is constructed by simply concatenating the constituent symbol models.

In order to model the lexicon, we use the efficient Trie (also known as prefix tree) representation. Each node in the tree corresponds to a symbol and the models corresponding to the common prefixes are shared across words. The Trie representation considerably reduces the recognition time in addition to having low space complexity.

During recognition, the standard Viterbi algorithm is used to determine the conditional probability of a handwritten word given the word model. The label associated with the word model that has the maximum probability is assigned to the test word.

6. Dataset Description

6.1. Tamil

The Tamil dataset includes word samples written normally on a Tablet PC by 132 writers. 112 writers' data was used for training and the rest for testing. In order to evaluate writer-independent recognition, no writer is present in both the train and test sets. All the samples in the dataset were manually cleaned and labeled at the symbol level. Out of the 981 word samples in the test set, 707 samples were written discretely i.e. there was a pen-up in between every two consecutive symbols. Currently, our recognition system supports only discrete style of writing. This is mainly because, when the stroke-level preprocessing is applied to a stroke that corresponds to two or more symbols written cursively, the feature values are likely to change significantly. The remaining 284 word samples had mixed (cursive and discrete) style of writing and hence were not used for evaluation. Details of the data collection process may be found in [2].

6.2. English

Isolated lowercase character samples from the IRONOFF [9] dataset were divided in the ratio 2:1 to form the train and test character sets. The train set was used to train the symbol models as described before. The test set was used to artificially generate 3000 word samples from the isolated character dataset for evaluating word recognition accuracy. This was possible because of stroke-level preprocessing and our assumption of discrete-style input. More details about the generation of the test set may be found in [1].

7. Experimental Evaluation

The recognition accuracy of the system for both English and Tamil was computed on lexicons of different sizes. Each *N*-word lexicon used the top *N* most frequently used words in the language derived from a text corpus. During evaluation, it was ensured that the sequence of symbol IDs corresponding to the test word was present in the lexicon in accordance with our standard writing order constraints. Table 1. compares the accuracies for normal writing (discrete and mixed-style) reported in [2], normal writing with only discrete style, and eyes-free writing for Tamil. It is encouraging to note that the accuracy on eyes-free input is only marginally worse than discrete despite discarding the position information. This suggests that for

Table 1. Recognition accuracy for normal and eyes-free handwriting for Tamil and English

Lexicon Size	*Tamil Word Recognition*				*English Lowercase Word Recognition*		
	Normal Writing [2]	Normal Writing (Discrete Input)	Eyes-free Writing		Normal Writing [5]	Eyes-freeWriting [1]	
			Top1	Top2		Top1	Top2
1K	97.96	97.88	96.75	97.17	-	94.87	96.4
2K	95.82	97.31	96.18	96.75	-	93.6	95.63
5K	94.49	96.18	95.19	96.18	96	91.97	94.43
10K	93.17	94.77	94.20	96.04	94.7	90.63	93.87
20K	92.15	93.49	92.93	95.33	93.4	89.17	93.17

Tamil, most of the information is in the shapes of strokes. It is also interesting to note that the accuracy obtained on eyes-free writing is more than that of the dataset that has both mixed and discrete style samples. One possible reason could be that the system for recognizing mixed style of writing is burdened with the task of segmentation. On the other hand, segmentation issue does not arise in the case of eyes-free handwriting as it assumes that the input word is written discretely. Table 1. also compares the accuracies of eyes-free and normal writing for English. One may note that the top 2 accuracy of eyes-free writing is close to the state-of-the-art accuracy for normal writing published in the literature [5]. The samples of lowercase letters in the IRONOFF [9] dataset are written cursively and as a result the similarity between symbols tends to be high. For instance, 'l' and 'e' appear quite similar when written in cursive style. This could possibly be the reason for the accuracy of eyes-free writing of Tamil being more than that of English. The experimental setup of our investigations for Tamil is summarized in Table 2.

Table 2. Experimental setup for Tamil

Symbol HMM	15 Gaussian components per state, variable number of states, left-to-right HMM with no state skipping
Lexicon	Trie representation
Viterbi Setting	Beam width = 750
Datasets	
Training:	112 writers, approx. 30 words, 2 samples per word per writer
Testing:	20 writers, 707 discrete-style word samples

8. Conclusions and Future Directions

In this paper, we described the notion of "eyes-free writing", our solution for continuous input on a small device. We then discussed the challenges for recognition of such input. We also described the different components of our HMM-based word recognition system. The stroke-level preprocessing was shown to help in discarding the unreliable position information resulting from eyes-free writing. The recognition accuracy for eyes-free Tamil writing was found to be comparable with normal writing and hence it is encouraging to pursue this research thread further.

Our immediate next step is to evaluate the performance of eyes-free handwriting recognition on other Indic scripts such as Devanagari and Telugu. Since the Devanagari script is considered to be more complex than Tamil, we expect a few challenges ahead. For instance, unlike in Tamil, Devanagari writing has a lot of stroke order/number variations as well as symbol order variations, in which case our writing order constraint may not be justified. We would also like to support mixed (discrete-cursive combination) styles of writing in our future system, which we believe may be achieved by selection of appropriate features. Along with the lexicon, one may also make use of language models to improve recognition performance.

9. References

[1] Bharath A. and Sriganesh Madhvanath. "FreePad: A Novel Handwriting-based Text Input for Pen and Touch Interfaces," *Proceedings of the International Conference on Intelligent User Interfaces*, Canary Islands, Spain, Jan 13-16 2008.

[2] Bharath A and Sriganesh Madhvanath. "Hidden Markov Models for Online Handwritten Tamil Word Recognition," *Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR)*, Curitiba, Brazil, 2007, pp. 506-510.

[3] F. Coulmas. *The Blackwell Encyclopedia of Writing Systems*. Blackwell, Oxford, 1996.

[4] D. Goldberg, and C. Richardson. "Touch-typing with a stylus," *Proceedings of the ACM Conference on Human Factors in Computing Systems - INTERCHI*, Amsterdam, The Netherlands, 1993, pp. 80-87.

[5] S. Jaeger, S. Manke, J. Reichert, and A. Waibel. *Online Handwriting Recognition: The NPen++ Recognizer*. International Journal on Document Analysis and Recognition, 3:169–180, 2001.

[6] M. Pastor, A. Toselli, and E. Vidal, "Writing Speed Normalization for On-Line Handwritten Text Recognition," *Proceedings of the 8th International Conference on Document Analysis and Recognition (ICDAR)*, Seoul, Korea, 2005, pp. 1131–1135.

[7] R. Rabiner "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of IEEE*, 1989, 79(2), pp. 257-286.

[8] H. Shimodaira, T. Sudo, M. Nakai, and S. Sagayama. "On-line Overlaid-Handwriting Recognition Based on Substroke HMMs," *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR)*, Edinburgh, Scotland, 2003, pp. 1043 – 1047.

[9] C. Viard-Gaudin, P. M. Lallican, P. Binter, and S. Knerr. "The IRESTE On/Off (IRONOFF) Dual Handwriting Database," *Proceedings of the Fifth International Conference on Document Analysis and Recognition (ICDAR),* Bangalore, India, 1999, pp. 455 – 458.