Voice Key Board: Multimodal Indic Text Input

Prasenjit Dey
Hewlett-Packard Labs
24, Salarpuria Arena
Adugodi, Hosur Road
Bangalore-560030, India
pdey@hp.com

Ramchandrula Sitaram
Hewlett-Packard Labs
24, Salarpuria Arena
Adugodi, Hosur Road
Bangalore-560030, India
sitaram@hp.com

Rahul Ajmera
Human Factors International
310/6, H.R Complex
Koramangala, 5th Block
Bangalore 560095, India
rahul@humanfactors.com

Kalika Bali
Microsoft Research India
"Scientia", 196/36 2nd Main
Sadashivnagar,
Bangalore-560080, India
kalikab@microsoft.com

ABSTRACT
Multimodal systems, incorporating more natural input modalities like speech, hand gesture, facial expression etc., can make human-computer-interaction more intuitive by drawing inspiration from spontaneous human-human-interaction. We present here a multimodal input device for Indic scripts called the Voice Key Board (*VKB*) which offers a simpler and more intuitive method for input of Indic scripts. *VKB* exploits the syllabic nature of Indic language scripts and exploits the user's mental model of Indic scripts wherein a base consonant character is modified by different vowel ligatures to represent the actual syllabic character. We also present a user evaluation result for *VKB* comparing it with the most common input method for the Devanagari script, the *InScript* keyboard. The results indicate a strong user preference for *VKB* in terms of input speed and learnability. Though *VKB* starts with a higher user error rate compared to *InScript*, the error rate drops by 55% by the end of the experiment, and the input speed of *VKB* is found to be 81% higher than *InScript*. Our user study results point to interesting research directions for the use of multiple natural modalities for Indic text input.

Categories and Subject Descriptors
H.5.2 [User Interfaces]: User interface management systems; Voice I/O; Natural language; D.2.2 [**Software Engineering**]: Design Tools and Techniques - *user interfaces;*

General Terms: Algorithms, Human Factors

Keywords: multimodal systems, text input, voice keyboard, human-computer-interaction, syllabic scripts, Indic text.

1. INTRODUCTION
The lack of a good system for Indic text input has been one of the primary barriers to penetration of computing systems in India. Keyboards for Indic text input have proven unnatural, and difficult to learn and use. Multimodal systems [1] which use multiple modalities like speech, touch and gestures for interaction, potentially require very little learning and are less intimidating because they use modalities which are natural to our everyday human communication. We are therefore motivated to explore the use of multiple modalities to address the problem of Indic text input.

Indic scripts such as Devanagari, used for Hindi and other Indic languages, are defined as "syllabic alphabets" in that the unit of encoding is a syllable, however the corresponding graphic units show distinctive internal structure and a constituent set of graphemes. The formative principles behind them may be summarized as follows [2]:

- graphemes for independent (initial) Vs (vowel)
- C (consonant) graphemes with inherent neutral vowel *a*
- V indication in non-initial position by means of *matras* (V diacritics)
- ligatures for C clusters
- muting of inherent V by means of a special diacritic called *virama*

Most Indic scripts have of the order of 600 CV units and as many as 20,000 CCV ones in theory, although only a much smaller subset (especially of CCV units) is used in practice. The V diacritics and ligatures for C clusters are not standardized in some scripts. Therefore, standard QWERTY keyboards as means of text input in Indic scripts such as Devanagari, Tamil etc., as well as other syllabic scripts like Sinhalese and Thai, face unique challenges due to script complexity and the sheer size of the syllable set. Complex keyboard layouts for Indic script input add to the already existing complexity and unfamiliarity of traditional desktop user interfaces, and further raise the barrier to adoption by lay users.

In this paper we describe the Voice Key Board (*VKB*). In order to enter a CV character, the user presses the key corresponding to a consonant C on the keyboard while simultaneously uttering the syllable CV in speech. This results in information from the two modalities (keyboard and speech) which are both supplementary and complementary in nature. The accurate recognition of C from the keyboard can be used to disambiguate recognition of the spoken CV syllable.

The paper is organized as follows. We review related Indic text input work in Section 2. We describe the Voice Key Board system in Section 3. Some comparisons with an existing system and user evaluation results are presented in Section 4. We conclude with discussion, summary and next steps, in Section 5 and Section 6.

ICMI-MLMI'09, November 2–4, 2009, Cambridge, MA, USA.

2. RELATED WORK

As described earlier, Indic scripts face an unique challenge due to script complexity, and the sheer size of the syllable set resulting from various CV and CCV combinations.

To accommodate such a large character set, current techniques employ either a keyboard where several key strokes may be required to enter a desired character, or a handwritten character-recognition technique that recognizes entire characters. For example, Figure. 1 shows the process of input of the character /kii/ in the Devanagari script used for Hindi by pressing two keys /ka/ (base character) and /ii/ (vowel modifier) in sequence. This QWERTY keyboard layout, called *InScript*, may require as many as 3-4 keystrokes to enter a single character representing conjuncts like /kta/, /pna/ etc. which are fairly frequent in syllabic languages such as Hindi, Tamil and Bengali [3].

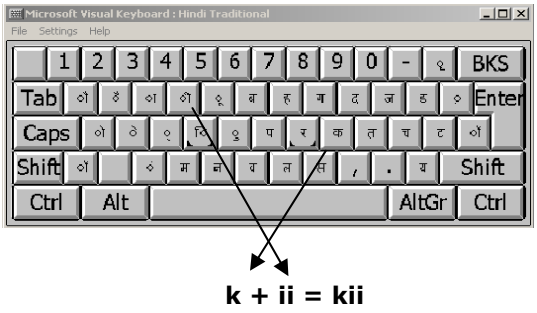

Figure 1: *Inscript Keyboard Layout*

Not only does the keyboard approach provide incomplete visibility of the entire character map at any given point of time but these keyboards are non-intuitive and can require an extensive practice period for achieving proficiency of use. Also, the increasing demand for smaller and smaller devices is driving keyboard design towards smaller, one-handed keypads that cannot accommodate large character sets required for Indic scripts. This poses a severe problem when handheld devices, such as PDAs and mobile phones, have to be used for input of data in these scripts.

Solutions using the handwritten character-recognition approach allow the user to write an entire character on a graphics tablet in a natural manner using a stylus. In this approach, a character-recognition engine attempts to recognize the character from the strokes entered on the tablet. However, current handwritten character-recognition techniques give significantly low recognition accuracies for Indic scripts due to their complex shapes, and the considerable variety in writing styles.

Some hybrid pen based techniques have also been proposed to improve the accuracy of Indic character recognition. For example *IndicDasher* [4] is very similar to the English Dasher [5] except that the vowel modifier or diacritic is specified as a pen gesture on the selected base character. This requires recognition of the digital ink only the vowel modifier and not of the base character. In the Gesture Key Board (*GKB*) [6], the base character is specified by pointing the pen at the location of the base character in a printed grid pasted on a tablet and the vowel modifier is specified by drawing the vowel modifier on top of the base character. As with *IndicDasher*, the base character recognition is avoided by using the pen location, and only the vowel modifiers drawn on top of the base characters are recognized.

The *T9* [7] input method has also been adapted for input of Indic text on mobile phones. This requires one or several key-presses to specify the base character, and another single or several key-presses on a different key to specify the vowel modifier until the desired syllable is displayed. The prediction mechanism displays a list of related words from which the desired word can be selected, or one can continue to type more characters to get more refined choices. Spelling out-of-vocabulary word requires toggling to complete typing mode from the predictive T9 mode.

Speech only solutions relying on speech recognition are also used for text entry as in a dictation systems. Though intuitive, this approach suffers from issues of recognition accuracy in other application scenarios. There are usability issues even with existing systems. Hesitations, interruptions, and background noise, for example, can lead to errors that are difficult to recover from, unlike keyboards. Further, development of commercial speech engines for Indic languages is still in its infancy.

3. VOICE KEY BOARD

Voice Key Board (*VKB*) is proposed as a simpler way to support input of Indic characters, by using voice to augment keystroke input. Here, the necessary modification of base letters for Indic scripts is achieved by simultaneously speaking out the syllable represented by the CV combination, as the base character C is being input either by pressing a key or by writing using a pen-based device (Figure. 2). Thus, entering /kii/ would now require a single key-stroke, that is, the user now has to press the key “k” and speak out “/kii/”. For a word like “bhaarat” (India), the number of keystrokes would reduce from six using a conventional QWERTY layout to just three. That is, instead of pressing two keys for “bh”, two for “aa”, and one each for “ra” and “t”, the user needs to press only three keys, one each for “bh”, “r” and “t” while speaking out the associated syllables. This method naturally uses the phonetic nature of Indic scripts where a one-to-one relationship exists between the character and the sound it represents.

The solution proposed here requires the integration of an Automatic Speech Recognizer (ASR) with the text-input method. The recognition vocabulary of the ASR is dynamically determined and is limited in size by the base character keyed in, resulting in faster and better recognition. For example, when /kii/ is entered by keying in the base letter “k” and speaking out “/kii/”, the active vocabulary of the ASR would be limited to /ka/, /kaa/, /ki/, /kii/, /ku/, etc., (~15 characters in number) that is, the combinations of the consonant “k” with various vowel modifiers. Thus, not only will the speech recognizer perform with higher accuracy due to the smaller vocabulary but also the instances of confusion between similar sounds are considerably reduced. The speech recognizer does not have to distinguish between difficult and mutually confusing sounds such as /pa/ vs. /ba/, /ma/ vs. /na/, /ka/ vs. /ga/ etc., as these sounds are mapped to and produced along with different base characters. The computational complexity of this kind of speech recognizer is also significantly less compared to traditional dictation systems. Further, this method is suitable for usage scenarios involving interruption, and

resumption of the entry at a later stage, thus preserving the advantages of a normal keyboard.

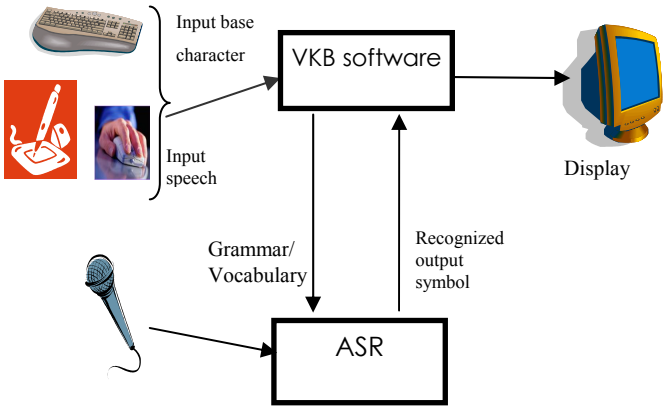

Figure 2: *VKB system architecture*

3.1 System Overview

We have developed a prototype of *VKB* that uses a soft keyboard as shown in Figure. 3. The GUI is script independent and uses the Unicode character encoding to support rendering of Indic scripts. The language can be changed dynamically by choosing from a drop-down list. The GUI consists of a soft keypad corresponding to the base-characters of the chosen language and a display box for rendering the final text entered by the user. For isolated vowels on the keypad, no speech input is needed and they are selected by tapping on the appropriate key for the vowel. For consonants, speech is used to specify the appropriate CV combination. The backend speech engine may be based on statistical recognition algorithms (e.g. Hidden Markov Models) or template matching algorithms (e.g. Dynamic Time Warping, DTW). For our application, we implemented a standalone DTW based speech engine as it is fast and works quite well for restricted domain, speaker dependent speech input. A trainer for the speech engine was also developed that takes user's speech input corresponding to different modifier combinations of all base-characters for creating the template inventory as shown in Figure. 4. These model templates are matched against the user's speech input to recognize the CV combination uttered by the user.

Temporal Integration: To enter a character the user first presses the corresponding base key on the keypad of the GUI and speaks the syllable CV combination. In case the key pressed is a vowel, speech input for a pre-determined duration. In case no speech input is given the system renders the base character without the modifier.

Semantic Integration: Our system performs early semantic integration [8] wherein once the base character and the speech modifier are entered, the ASR interface is invoked and an ASR vocabulary is dynamically created corresponding to the various CV combinations for the base C entered by the user using the keyboard. This vocabulary along with the recorded speech is sent to the ASR engine which performs the recognition and returns the recognized CV character, which is rendered in the display box. the character is rendered as it is. Else the system waits for user's The application works in real-time. The prototype currently supports five Indic languages and the same ASR engine is used across all of them as the sound of the CV combinations are same for all these languages, unlike their scripts which are all different.

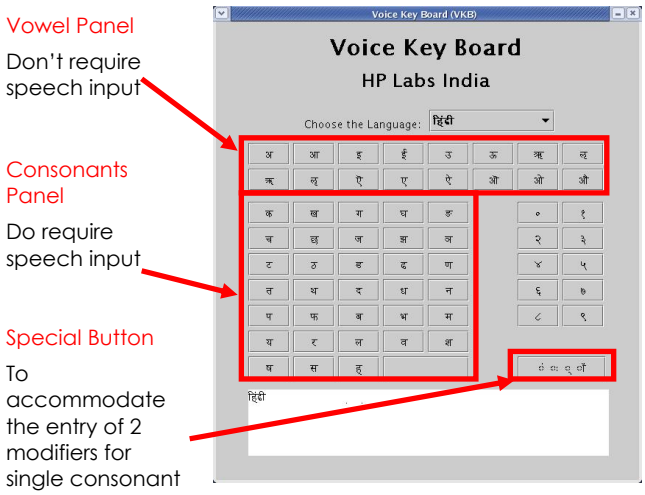

Figure 3: *Voice Keyboard Layout (for Hindi language)*

Figure 4: *Devanagari (Hindi) voice keyboard trainer*

4. EVALUATION

4.1 Goals

Our goal for user evaluation was to compare the performance of *VKB* with *InScript* keyboard [3] for the entry of Devanagari script. We also sought to collect subjective satisfaction data to make a holistic assessment of the proposed technique. We chose *InScript* as the technique for comparison because it is the most prevalent keyboard based input technique used for Devanagari script, and a common baseline for comparison of other techniques that have been proposed [4, 6].

4.2 Apparatus

For the implemented *VKB* system for Devanagari (Hindi) characters, the speech syllable recognition accuracies obtained after training were only of the order of 90%. For user evaluation, we have different users who are being tested in quick succession – a speaker independent situation in which the speech recognizer may commit many errors in recognition of syllables. To be able to compare the efficacy of the *VKB* method without contamination from system errors, and in order not to unnecessarily frustrate users, we used a Wizard-of-Oz technique for syllable recognition

in *VKB*. This allowed separation of system errors from user errors for analysis, as opposed to looking at an overall error figure which may be difficult to analyse. While this is not optimal, we believe that this realistically simulates the eventual scenario involving speaker-dependent *VKB* for personal use.

As already mentioned, we used an *InScript* keyboard for input of Devanagari script for the purpose of comparison.

4.3 Participants

Twenty participants volunteered for the experiment. They were recruited from within the office environment. They were from the age group 20-33 years, and consisted of an almost equal mix of men and women. Participants received compensation in the form of a gift certificate of nominal value.

4.4 Experimental Procedure

The performance of the techniques was measured as the number of words entered per 10 mins. Participants were given two blocks of Devanagari text of 150 and 156 words each to enter the words from. The first block of text was entered by participants for 10 mins after which there was a resting time of 5 mins, and then the second block of text of 156 words was given to the participants to enter for 10 mins. Two successive blocks of text were given to the participants to evaluate the learnability of the technique, which manifests as improvement in performance while entering the second block of text. Each block was timed from when the first character was typed in to when 10 mins were completed.

Before beginning the experiment, sufficient training of the techniques was provided to the participants as follows:

InScript: A video demonstration of the technique was provided to the participants for using the *InScript* keyboard. The participants were then given a block of text (different from the two blocks of text used during the actual experiments) to enter using the technique for 5 mins. Sufficient demonstrations were given by the experimenter to make sure the technique was understood properly by the participants.

VKB: A live demonstration of the technique by the experimenter was provided to the participants. The participants were then given a block of text (different from the two blocks of text used during the actual experiments) to enter using the technique for 5 mins. Sufficient demonstrations were given by the experimenter to make sure the technique was understood properly by the participants.

After entering the two blocks of text, a feedback session was conducted by the experimenter for the purpose of subjective evaluation. A questionnaire with subjective evaluation of learnability, efficiency, efficacy and satisfaction was given to the participants to be evaluated on a scale of 1 to 7, indicating strong disagreement and strong agreement respectively. The experimenter also had a post-experiment interview with the participants to capture any other subjective data that may have been missed.

For *InScript*, after each experiment, the number of words entered per 10 mins from each block of text was manually tabulated by the experimenter. The word errors in the entered text were also manually tabulated by the experimenter. For *VKB* too, after each experiment, the number of words entered per 10 mins from each block of text was manually tabulated by the experimenter. However, the word errors for *VKB* were provided automatically by the interface software.

4.5 Design

A within-subjects design was used where participants were randomly assigned to two groups of 10 participants each. The first group performed the experiment with *VKB* first and then with *InScript*, and the other group did these in the reverse order. For each group, at least 24 hours elapsed between using each of the techniques to limit interference.

4.6 Results

4.6.1 Performance

Figure 5 shows the analysis of the data collected for the 20 participants for typing in 2 blocks of text for 10 mins in succession, with a gap of 5 mins between blocks. The average number of words that are input in 10 mins for *InScript* is 45.32, and for *VKB* it is 82.18, which is about 81% higher compared to *InScript*. This clearly points to a superior performance of the *VKB* technique compared to *InScript*. Analysis of variance also indicated significant effect of technique on the word input rate ($F_{1,78} = 31.5 > F_{critic}=16.8$, $p < 0.0001$).

If we look at the block by block performance, in case of *InScript*, the average word input rate goes from 44.85 for Block-1of text to 45.8 for Block-2 of text, which represents a 2 % increase. In the case of *VKB,* the average word input rate goes from 75.95 for Block-1of text to 88.4 for Block-2 of text. This approximately 16% increase underscores the learnability of *VKB*, and increased proficiency with use.

Figure 6 shows the number of word errors in the input text. For *InScript*, the mean error for Block-1 of text is 4.05 and goes to 4.9 for Block-2 of text, which indicates that there is no significant change in the error rate from one block of text to the next.

For *VKB*, the mean error for Block-1 of text is 11.3 and goes down to 5.05 for Block-2 of text, which indicates a significant drop in the error rate of 55 % from one block of text to the next. This shows that although *VKB* starts out performing worse than *InScript* initially, by the end of the second block of text, the error rate for *VKB* drops significantly to a level comparable to *InScript*. This significant drop in error levels, coupled with the simultaneous and significant increase in input speed for *VKB* by the end of Block-2, indicates the strong effect of learning on the performance of *VKB*.

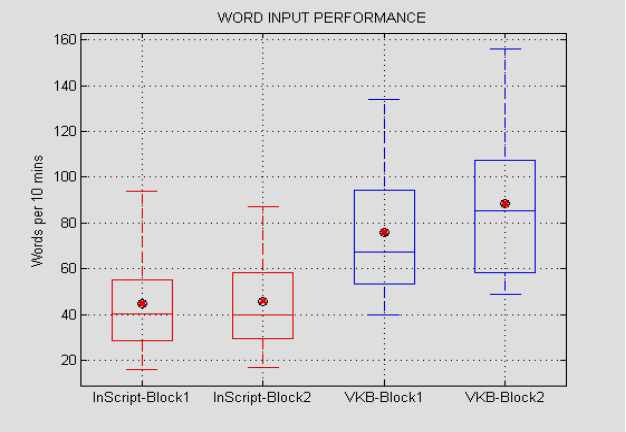

Figure 5: *Box plot of the number of words input per 10 mins for the VKB and InScript techniques for each block of text. The mean value is indicated by the red circle.*

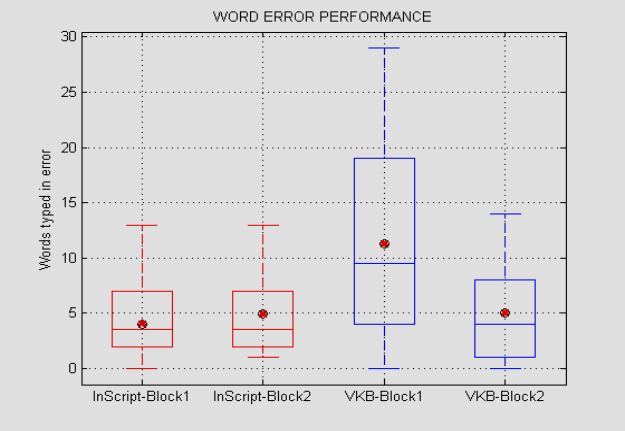

Figure 6: *Box plot of the number of words input in error per 10 mins for the VKB and InScript techniques for each block of text. The mean value is indicated by the red circle.*

4.6.2 Subjective Satisfaction

The subjective satisfaction questionnaire had statements to measure learnability, satisfaction, efficiency and efficacy. Each of these vectors was measured with the help of two to three different statements. The ratings on the statements on efficiency reinforced the data we obtained from the performance tests − the participants felt *VKB* to be a more efficient input method. The other significant finding was that the learnability of *VKB* was perceived to be much higher than that of *InScript*. While the ratings for satisfaction and efficacy were higher for *VKB* than *InScript*, the difference in ratings was not found to be statistically significant.

The entire session was followed by a brief open discussion with the participants. Some of the interesting observations from the discussion are:

1. Many participants felt that they would not like to use the *VKB* in office environments since others around them would get disturbed and also would get to know what they are typing.
2. While they were instructed to speak out syllable by syllable, many ended up speaking the entire word. This they felt was much more intuitive.
3. In case of errors in *VKB*, the participants often ended up repeating the syllable more loudly.
4. While the participants were instructed to say the syllable before pressing the related character (to simplify the Wizard-of-Oz implementation), participants mentioned that it felt better to say it together with the key-press and sometimes even after it.

5. DISCUSSION

In our experiments, *VKB* came out very strongly in terms of efficiency and learnability as compared to *InScript*. This may be attributed to its matching the user's mental model for writing Indic scripts where the base consonant is written first and then the vowel modifier is added to it. For *VKB*, the acceleration in word input rate, and the significant reduction in word error rate from one block of text to the next, points to phase of rapid learning that the user goes through when starting to use *VKB*. This acceleration of input speed may have started even before the ending of Block-1 but we did not have data at the sub-block granularity to ascertain this. Though the performance of *VKB* is far superior to *InScript*, issues of adoption like use of speech in public or office environments still need to be addressed.

In our user evaluation, we used a Wizard-of-Oz technique and not an ASR for syllable recognition. Our main aim was to analyse and understand user satisfaction and performance for the *VKB* as a technique. Had we used an ASR based system, the user evaluation data would have reflected performance and satisfaction issues resulting from both system errors and user errors. An analysis of this data would not have lead to a clear understanding about the technique itself from a user standpoint. Given that speech recognition technology is rapidly maturing, we believe that for our limited vocabulary of syllables (syllables only for the C pressed on the keyboard), speaker-dependent recognition performance is likely to exceed 99%. This is also in line with the most likely usage scenario for *VKB*, which we believe will be deployed on systems for personal use.

6. SUMMARY AND NEXT STEPS

The keyboard has been a very reliable input mechanism for many years. However, inspite of its reliability, it has not made computing easier for the masses in the Indian context because of its complexity for Indic scripts with their large character sets. Multimodal systems which combine a reliable modality for input like keyboard, with a more natural input modality like speech help move the computing experience a step closer to the ideal of natural human-human interactions. Such designs also help overcome the unreliability of recognizers for natural human input modalities by exploiting the property of mutual disambiguation.

We have described *VKB*, a new technique for Indic text input, which exploits some of the aforementioned properties of multimodal systems. *VKB* combines the keyboard modality with speech input to allow more naturalness as compared to purely keyboard-based text input, while leveraging the user's mental model of how Indic scripts are actually written. Our initial experiments show that *VKB* significantly outperforms *InScript* in terms of input speed and learnability.

In the future we plan to improve the recognition accuracy of syllable recognition by replacing our DTW-based engine with speaker dependent models in a commercial speech recognition engine, and evaluate the system using ASR based syllable recognition. We would also like to explore the use of n-gram language models to improve syllable recognition accuracy. From an acceptance perspective, it would be useful to examine user behavior with *VKB* in different environments. We would also like to examine more deeply the use of *VKB* for other device form factors such as mobile phones, and the design of interactions for such form factors.

7. ACKNOWLEDGMENTS

We would like to thank all those who participated in the *VKB* evaluation experiments for their valuable feedback. We would also like to thank Sriganesh Madhvanath at HP Labs for his valuable inputs and discussions during the course of preparation of this paper. Finally, we would like to thank our anonymous reviewers for their critical comments and suggestions which greatly helped improve the contents of this paper.

8. REFERENCES

[1] Oviatt, S. L., "Multimodal interfaces, Handbook of Human-Computer Interaction (revised edition)", (ed. by J. Jacko & A. Sears), *Lawrence Erlbaum Assoc*.,Mahwah, New Jersey, 2006.

[2] Coulmas, Florian, *The Encyclopedia of Writing Systems*, Blackwell Publishing, 1999.

[3] InScript: http://en.wikipedia.org/wiki/InScript_Typing

[4] Srinivas N K, Nobby Varghese, RKVS Raman, "Indic-Dasher : A Stroke and Gesture Based Input Mechanism for Indic Scripts", 2008. IUI4DR: intelligent user interfaces for developing regions. In *Proceedings of the 13th international Conference on intelligent User interfaces* (Gran Canaria, Spain, January 13 - 16, 2008). IUI '08. ACM, New York, NY, 437-437

[5] Dasher: http://www.inference.phy.cam.ac.uk/dasher/

[6] R. Balaji, V. Deepu, Sriganesh Madhvanath and Jayasree Prabhakaran. "Handwritten Gesture Recognition for Gesture Keyboard", *Proceedings of the 10th International Workshop on Frontiers in Handwriting Recognition (IWFHR10)*, 2006.

[7] T9: http://www.t9.com/us/ask.asp

[8] Johnston, M. and Bangalore, S. 2005. Finite-state multimodal integration and understanding. *Nat. Lang. Eng.* 11, 2 (Jun. 2005), 159-187.