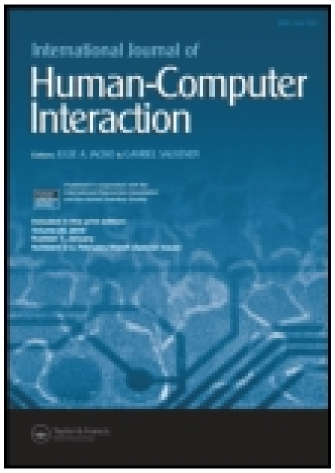


International Journal of Human-Computer Interaction
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/hihc20

An Approach to Design Virtual Keyboards for Text Composition in Indian Languages

Debasis Samanta [a] , Sayan Sarcar [a] & Soumalya Ghosh [a]
[a] Indian Institute of Technology Kharagpur , Kharagpur , India

Accepted author version posted online: 19 Sep 2012.Published online: 10 Jun 2013.

To cite this article: Debasis Samanta , Sayan Sarcar & Soumalya Ghosh (2013) An Approach to Design Virtual Keyboards for Text Composition in Indian Languages, International Journal of Human-Computer Interaction, 29:8, 516-540, DOI: 10.1080/10447318.2012.728483

To link to this article: http://dx.doi.org/10.1080/10447318.2012.728483

Intl. Journal of Human–Computer Interaction, 29: 516–540, 2013
Copyright © Taylor & Francis Group, LLC
ISSN: 1044-7318 print / 1532-7590 online
DOI: 10.1080/10447318.2012.728483

An Approach to Design Virtual Keyboards for Text Composition in Indian Languages

Debasis Samanta, Sayan Sarcar, and Soumalya Ghosh
Indian Institute of Technology Kharagpur, Kharagpur, India

Of late there has been significant development in Information and Communication Technology (ICT), which offers interaction with computing systems in a large scale. Text input mechanisms in users' own languages are necessary for bringing the ICT advantages to the English illiterates. *QWERTY* keyboard, however, which was designed for text entry in English, is not as suitable for text composition in other languages. As an alternative, researchers advocate virtual keyboards in users' mother languages. This article proposes an approach to designing virtual keyboards suitable for text entry in Indian languages. Composition of texts in Indian languages using virtual keyboards needs special attention due to the presence of large character sets, complex characters, inflexions, and so on. First, we examine the suitability of existing design principles in developing virtual keyboards in Indian languages. Then we propose a virtual keyboard layout suitable for efficient text entry in Indian languages. We have tested our approach with the three most spoken languages in India, namely, Bengali, Hindi, and Telugu. Performance of the keyboards have been evaluated, and the evaluation substantiates that proposed design achieves on average 21.3% higher text entry rather than with conventional virtual keyboards. The proposed approach, in fact, provides a solution to deal with complexity in Indian languages and can be extended to many other languages in the world.

1. INTRODUCTION

Presently, people all around the world communicate using handheld digital devices like cell phones and PDAs as well as desktop PCs and laptops. Recent statistics reveals that nearly 1 billion text messages are sent each day (MacKenzie & Tanaka-Ishii, 2007). In an Indian context, message composition draws more attention, where 220 million mobile subscribers sent more than a billion text messages in 2007–2008 (Newslog, 2010). According to the report of Technology Market Research (Reports, 2009), there were 1.4 billion e-mail users in 2009. They also reported that approximately 247 billion e-mails were sent worldwide each day in 2009. It is predicted that this figure will almost double to 507 billion messages per day by 2013, with the expected increment of users up to 1.9 billion (Releases, 2009). In spite of the speedy growth of the text composition task, text composition rate is still too low. Text composition rate of expert typist users in the English language is 30 to 45 words per minute (Brown, 1988), whereas for computer users having basic knowledge it is only 5 to 10 words per minute (Fazly, 2002). The result becomes even lower in the case of an Indian language like Hindi, Bengali, or Telugu. Therefore, it is necessary to support a text composition task in digital devices and help the users to perform tasks with better text entry methodologies in coming years. As an example, in English, a speech-based text composition system augmented with error minimization support has already been developed (Fischer, Price, & Sears, 2006). A virtual keyboard is one of the solutions for addressing the requirement of providing a better text composition rate. It is a soft keyboard that imitates the QWERTY hardware keyboard ambiance for a text entry system. In addition, the configuration of a virtual keyboard can be changed effortlessly at any point of time, are which is not easily possible on a QWERTY keyboard. Also, virtual keyboards are beneficial because they can be implemented in any type of digital device with any language specification. In these devices, a text entry task is not possible through a conventional hardware keyboard due to mobility restriction, size factor, and so on.

These days, many applications in a wide range of digital devices like cell phones (Sears & Zha, 2003), iPhones, digital TVs (Perrinet et al., 2011), and so on, use a virtual keyboard to enter data or text. The text entry performance of a virtual keyboard depends on its design. The English-language-compatible virtual keyboards are based on some design strategies like the special arrangement of keys, a minimum distance between character pairs, and so on. A well-known key arrangement is known to be in QWERTY (Chubon & Hester, 1988, Sarcar et al., 2010) fashion. The principle behind designing another popular keyboard layout called FITALY (Fitaly, 2009) is the character frequency of the English alphabet, which decides the position of different character keys on the virtual keyboard. In FITALY, the frequently occurring keys are placed in

This work was carried out under a project sponsored by Department of Information Technology, Government of India. We express our sincere thanks to all participants for their painstaking efforts in the evaluation of keyboards.

Address correspondence to Sayan Sarcar, School of Information Technology, Indian Institute of Technology Kharagpur, Takshashila Building, Kharagpur, West Bengal 721302, India. E-mail: mailtosayan@gmail.com

the central position. The Space character, which is accessed most, has two buttons. MacKenzie and Zhang (1999a) proposed an optimized keyboard arrangement called *OPTI*. This layout requires four keys for Space, and other keys are spatially arranged based on their digraph probabilities (Jelinek, 1990). The characters having superior digraph probabilities can be arranged in the middle of the keyboard, and inferior probability valued character pairs are placed at the outer part of the keyboard. Zhai, Hunter, and Smith (2000) used physical laws to develop virtual keyboards. One of the approaches used mechanical simulation of a mess of springs of Hooke's Law (n.d.), which tries to minimize the distance between character pairs. *Hooke's keyboard* (Zhai et al., 2000) uses digraph probabilities between two characters of English alphabets and arranges the keys having higher value side by side followed by less probable characters to the other vacant space. In another approach, developers try to minimize the distance between strongly associated pairs and call the keyboard arrangement *Lewis* keyboard (Lewis, Kennedy, & LaLomia, 1999). Mouse movement time can further be minimized by text composition methodology called *Metropolis algorithm* proposed by Zhai et al. (2000). They applied the algorithm in developing the *Metropolis keyboard* interface suitable for desktop computing systems and Alphabetically Tuned and Optimized Mobi Interface Keyboard (ATOMIK; Zhai, Hunter, & Smith, 2002) layout useful for mobile systems. Most of the virtual keyboard design approaches in the English language use basic methodologies of language-processing tasks like language modeling (unigram and bigram; Manning & Schutze, 2002) from a corpus.

Existing design principles may yield better virtual keyboards for text entry in English, but it is reported that they are unable to achieve better text entry rate in Indian languages (Sarcar et al., 2010). The reasons for having inferior text entry rates can be summarized as follows.

- The features in Indian languages are completely different from English. In English, alphabets can be composed in uppercase or lowercase, whereas complex characters (*yuktakshar*; "Bengali Script," n.d.; "Devanagari," n.d.; "Telugu Script," n.d.), inflexion (dependent vowel), ("Languages of India," n.d.) and so on, are present in Indian languages. Moreover, the formation of a complex character requires at least three key presses, which is more with respect to English character composition (one for a single small character and an extra *shift* to form capital character). As a result, the user spends more time and effort to compose same-length characters in an Indian language as in English.
- The English alphabet consists of 26 characters. On the other hand, number of alphabets in Indian languages is 45 to 55. In addition to this, approximately 150 complex characters are possible. So, deciding an optimal key arrangements leads to an exploration of a large solution space and appears to be a difficult problem in the case of Indian languages.

A few virtual keyboards exist to compose texts in Indian languages such as Google (Google, 2009), Microsoft (Microsoft, 2009), gate2home (gate2home, 2010), LooKeys (LooKeys, 2009), Lipik (Lipik, 2007), MyLanguage (MyLanguage, 2008), Avro (OmicronLab, 2010), and so on. These keyboards follow alphabetical ordering or Inscript (Indian script) mapping technique (Technology Development for Indian Languages, 1986). Moreover, the existing virtual keyboards occupy a larger screen area.

We have addressed the issues of designing virtual keyboards for Indian languages in this article. We propose to retain good design principles of a English language-based virtual keyboard in our design. This guides us to put the merits of existing virtual keyboards into the proposed one. Next, an optimum arrangement of keys has been made. This enables us to reduce eye and mouse movements and thus improve text entry rate. We then augment the optimized keyboard with a strategy that deals with inflexions to reduce further mouse movements. Based on our approach, we propose a generic structure (named as *iLiPi*) that has been mapped to three Indian languages, namely, Hindi, Bengali, and Telugu. To substantiate the efficacy of our approach, we have conducted user evaluation and model-based evaluation. A schematic diagram showing different steps involved in our work is shown in Figure 1.

The rest of the article is organized as follows. In section 2, we discuss the existing keyboard design approaches. A comparison of the performances of existing keyboards is also presented in section 2. Effectiveness of the existing approaches toward the design of a virtual keyboard for text composition in Indian languages is discussed in section 3. Further, issues that are dealt with while designing virtual keyboards in Indian languages are highlighted in this section. Our proposed layout of a virtual keyboard is presented in section 4. Experimental results and analysis of our proposed design is presented in section 5. Finally, section 6 concludes the article.

2. EXISTING APPROACHES TO DESIGNING VIRTUAL KEYBOARDS

A number of approaches to designing a virtual keyboard have been reported in recent literature. In this section, the design principles of these keyboards are discussed.

2.1. Dvorak Keyboard

Dvorak, Merrick, Dealey, and Ford (1936) proposed an arrangement for the keys (Figure 2a) in a *Simplified Keyboard*, which dissimilates the design of the *QWERTY* layout through an analysis of the relative frequency of used letters. It attempts to minimize hand and finger movements (Shieh & Lin, 1999). The layout is designed for achieving an arrangement that supports more work for the right hand. Moreover, the layout distributes

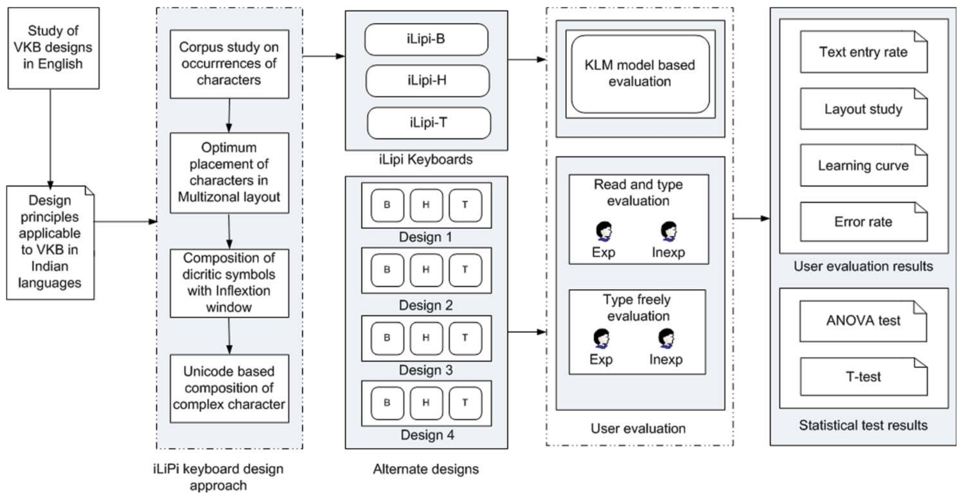

Study of VKB designs in English
Design principles applicable to VKB in Indian languages
Corpus study on occurrrences of characters
Optimum placement of characters in Multizonal layout
Composition of dicritic symbols with Inflextion window
Unicode based composition of complex character
iLiPi keyboard design approach
iLipi-B
iLipi-H
iLipi-T
iLipi Keyboards
Design 1
Design 2
Design 3
Design 4
Alternate designs
KLM model based evaluation
Read and type evaluation
Type freely evaluation
Exp
Inexp
User evaluation
Text entry rate
Layout study
Learning curve
Error rate
User evaluation results
ANOVA test
T-test
Statistical test results

FIG. 1. Work flow of iLiPi keyboards design approach. *iLiPi-B*, *iLiPi-H* and *iLiPi-T* denote virtual keyboards (VKBs) in Bengali (B), Hindi (H) and Telugu (T) languages, respectively.

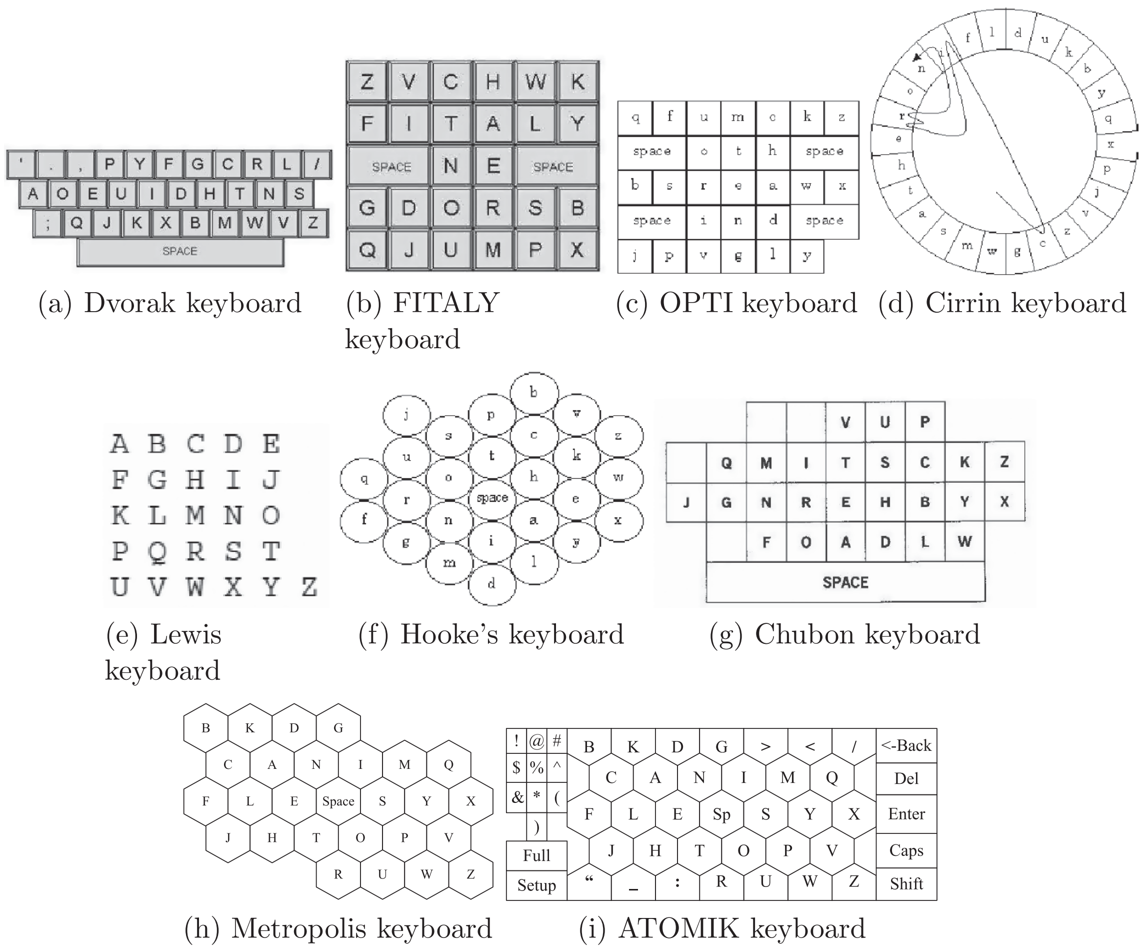


FIG. 2. Popular virtual keyboards in English. (a) Dvorak keyboard. (b) FITALY keyboard. (c) OPTI keyboard. (d) Cirrin keyboard. (e) Lewis keyboard. (f) Hooke's keyboard. (g) Chubon keyboard. (h) Metropolis keyboard. (i) ATOMIK keyboard.

letters based on the strength of each finger, keeping most frequently used letters on the home row. The design principle also includes placing vowels and frequently used consonants so that they can be typed with the alternate hand. The Dvorak virtual keyboard is designed mainly to optimize two-handed touch typing.

Later, the design was adopted to develop a mouse-based text entry system instead of its preliminary proposition as a hardware keyboard layout. With the hardware layout, the distance between the two most probable characters is within the moderate range of fingers of two opposing hands placed on a keyboard. There is considerably more mouse movement while composing text. For user study, the design has been chosen to represent QWERTY layout class.

2.2. FITALY Keyboard

The *FITALY One-Finger* keyboard (Fitaly, 2009) is designed to optimize mouse movements during text entry with one finger, a stylus, or a pen (Figure 2b). The most important feature of FITALY layout is the presence of two space bars. The proximity of the most common letters in English (e.g., E, T, A, H) to the space bars is also immediately apparent. The keyboard's name is taken from the letter sequence along the second row of keys. The average mouse movement is 1.8 keys compared to 3.2 keys for a QWERTY (Sholes et al., 1868) layout. These figures are obtained using a corpus of digraph probabilities similar to that described by Soukoreff and MacKenzie (1995). The layout of FITALY keyboard has been modified to single handed (right handed, left handed, or mouse driven) and double handed (in contrast to QWERTY layout) to increase the typing speed.

2.3. OPTI Keyboard

MacKenzie and Zhang (1999a) proposed one of the optimized virtual keyboard layouts for the English language named *OPTI* (Figure 2c). The keyboard layout is optimized to increase the typing speed using a trial-and-error method, Fitts' law (Fitts, 1954), and digram frequencies of characters in English vocabulary. Fitts' law gives a function for computing the key-tapping time given the length of the movement and width of the target. OPTI keyboard (MacKenzie & Zhang, 1999a) is theoretically 35% faster than QWERTY and 5% faster than FITALY layout. The error rate with OPTI (MacKenzie & Zhang, 1999a) is reported to be lower than the QWERTY keyboard.

2.4. Cirrin Keyboard

Figure 2d shows the layout of *Cirrin*, a text input method for a stylus-based user interface, proposed by Mankoff and Abowd (1998; word-level unistroke soft keyboard). Input is generated from the coordinates of the points where the stylus or pen crosses the circumference of the inner circle in the layout (see Figure 2d, e.g., composing the word *cirrin*). Mankoff and Abowd used a regular QWERTY keyboard and discussed using a regular handwriting recognizer for the task. The input mechanism has actually developed to facilitate stylus-based text entry. We extend the design for soft keyboard to facilitate mouse-based text entry.

2.5. Lewis Keyboard

Lewis et al. (1999) proposed a 5×6 virtual keyboard (Figure 2e) with an alphabetical sequence. It is a grouped alphabetic typing-key layout that claims to perform better than QWERTY layout for single-finger entry.

2.6. Hooke's Keyboard

Hooke's keyboard (Figure 2f) proposed by Zhai et al. (2002) uses the concept of Hooke's law in the design. The spring connects every pair of keys on the keyboard. The elasticity of a spring is proportional to the digraph probability between the keys. In equilibrium, the system may not be at a minimum energy state because some keys could block others from entering a lower energy state. At the end of each simulation, a minimal energy state is achieved manually to form a virtual keyboard layout. The procedure is repeated until a satisfactory layout had been formed.

2.7. Chubon Keyboard

Figure 2g shows the Chubon keyboard layout (Chubon & Hester, 1988). Chubon keyboard is designed to accelerate text entry for single-digit typists with a stylus. It is designed by centralizing the most commonly used letters in English language and placing frequently used letters in close vicinity to each other.

2.8. Metropolis Keyboard

Metropolis algorithm has been proposed by Zhai et al. (2000) to minimize energy (tension) while designing Hooke's keyboard. The algorithm is a Monte Carlo method widely used in searching for the minimum energy state in statistical physics. The high-performance keyboard design problem can be mapped into searching for the structure of a molecule (keyboard) at a stable low-energy state determined by the interactions among all the atoms (keys). Their approach follows *Random walk* in the virtual keyboard design space. In each step, the algorithm picks a character key and moves it in a random direction by a random amount to create a new configuration followed by evaluating Fitts' law. After that, whether the new configuration is kept as the starting position for the next iteration depends on the Metropolis algorithm. Developers replace the key circle shapes with hexagons in the design, which cover up the gaps and make more efficient use of the total space (see Figure 2h).

2.9. ATOMIK Keyboard

ATOMIK (Zhai et al., 2002, Figure 2i) is designed to address the problems with the QWERTY keyboard and to get better text entry rate in single-finger typing. The QWERTY keyboard remains unsuitable for keyboards with stylus-based text entry such as with iPod and PDA. This design is in fact good for two-handed typing with facilitation of frequent alternation of the left and right hand. While using a stylus keyboard, the stylus has to travel back and forth more frequently as well as over longer distance unnecessarily. ATOMIK achieves better movement efficiency than any other existing stylus keyboard in terms of stylus typing by accomplishing Metropolis or OPTI keyboard. The layout is alphabetically tuned in that letters from A to Z are placed from the upper left corner to the lower right corner of the keyboard. The arrangement mitigates the mental load of novice users in finding keys.

2.10. Other Virtual Keyboards for Text Entry in Indian Languages

Few interfaces have been proposed for composing text in Indian languages. Earlier, keyboards were designed by replacing the keys of hardware English keyboard with Indian language compatible characters. Joshi, Ganu, Chand, Parmar, and Mathur (2004) designed a *Devanagari Indic* script keyboard embedded in hardware in 2004 (Figure 3c). After observing the virtual keyboard design methodologies over two decades, it has been concluded that, apart from a few recently proposed designs, people have usually followed an *inscript* layout of keys in soft keyboards. The organization of the character keys is in the following manner:

- Vowels and consonants are separated. Vowels are placed on the left side of the keyboard, and the consonants are on the right.
- Characters with phonetic similarity are placed on the same key, making them easier to find. For example, the same key is used for the characters ক (k) and খ (kh) in Bengali. ক (k) is accessed in normal mode, whereas খ (kh) is written with the Shift key pressed. Similarly, the independent vowel for আ (aa) and the vowel sign for া (aa) are assigned to the same key.

Earlier, the Centre for Development of Advanced Computing has developed an *inscript* (Indian script) text entry software called *iLeap* (Technology Development for Indian Languages, 2009) to compose text in different Indian languages (Figure 3b). Further, Microsoft (2009) also designed an inscript on-screen virtual keyboard that supports text entry task in Hindi as well as other Indian languages ("Devnagari," n.d.; Figure 3e). Many other commercial text composition interfaces such as gate2home (gate2home, 2010), (Figure 3j), Lipik (Lipik, 2007); (Figure 3g), MyLanguage (MyLanguage, 2008), (Figure 3d), LooKeys (LooKeys, 2009); (Figure 3h) and so on, following incript design have been proposed in recent times. Further analysis reveals that in some designs, two characters (normal and Shift key tapped) are displayed on a single key (iLeap and Lipik), whereas others show one character on a key-based layout instance at a time. For later design, while tapping the Shift key on keyboard takes place, the alternate design with other characters on keys is displayed (microsoft, *gate2home*, *MyLanguage* etc.).

Google also launched a virtual keyboard (Google, 2010) to facilitate text-based query generation (Figure 3a) for Indian and foreign languages. Google (2009) and Guruji (2009; Figure 3i) have developed an Indic alphabetical keyboard to support users in composing texts in Indian language in order to reduce text composition time. Also, another alphabetical virtual keyboard in Bengali language, namely *Avro* (OmicronLab, 2010; Figure 3f), draws more attention to the user as the user feels more comfortable with composing text with the keyboard. Recently, some work on mobile-based text entry mechanisms has also been proposed (Mistry & Nayak, 2007).

To understand the impact of English language keyboard design principles on Indian language keyboards along with existing Indian language keyboards and analyze the human acceptance on those layouts, we perform expert user-based evaluation (Dix, Finley, Abowd, & Beale, 2004) in three Indian languages, namely, Hindi, Bengali, and Telugu. The evaluations have been conducted by two or three persons from the developers team along with expert users (UG/PG students) who participate in user evaluation. The details about the participants, apparatus, and evaluation procedures are mentioned in section 5. The expert user evaluation results, both for existing English language-based and Indian language virtual keyboards, are shown in Table 1 and 2, respectively.

3. LIMITATIONS AND ISSUES WITH EXISTING VIRTUAL KEYBOARDS

A preliminary investigation has been made to assess the efficacy of the afore mentioned virtual keyboards for composing texts in Indian languages. Our observations are summarized in this section.

3.1. Lower Text Entry Rate

It has been reported that the lower text entry rates (three to six words per minute) have been observed for Indian languages compared to their English counterparts (20 to 43 words per minute; MacKenzie & Zhang, 1999b, Sarcar et al., 2010).

3.2. Lesser User Friendliness

Virtual keyboard design principles support single finger or mouse-based text entry. The size of any Indian language-based virtual keyboard is larger than an English language keyboards, as the number of characters in any Indian language is moderately high. As the size of the keyboard increases, mouse and eye movements increase significantly compared to English virtual

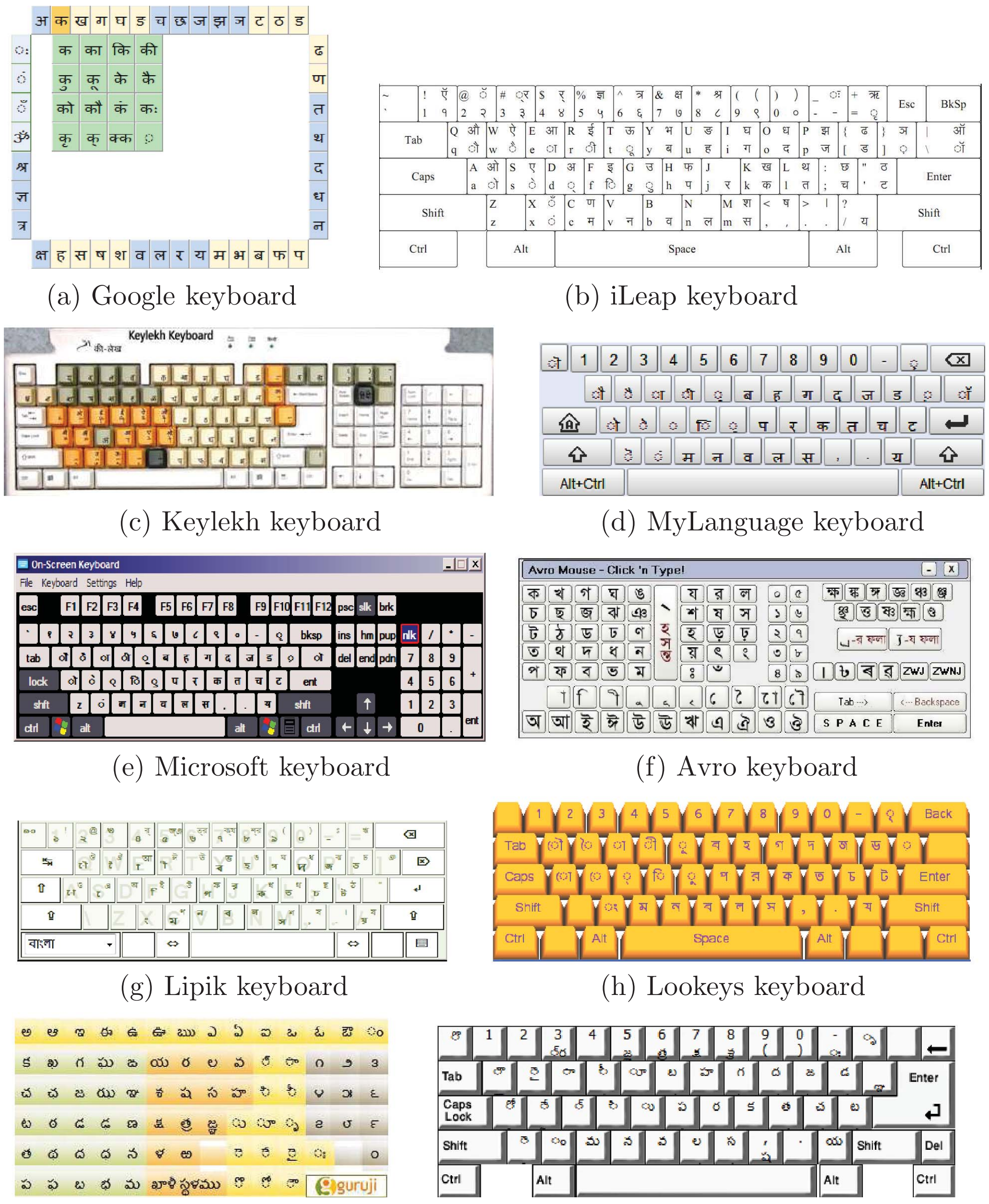


FIG. 3. Existing virtual keyboards in Indian languages. (a) Google keyboard. (b) iLeap keyboard. (c) Keylekh keyboard. (d) MyLanguage keyboard. (e) Microsoft keyboard. (f) Avro keyboard. (g) Lipik keyboard. (h) Lookeys keyboard. (i) Guruji keyboard. (j) gate2home keyboard.

keyboards. Moreover, working with a keyboard with a higher number of characters demands a higher cognitive load of the user.

3.3. Error Prone Text Entry

It is frequent for the majority of users to commit errors. Also, there are many more errors committed by users in Indian language-based text entry than in English language text composition because of the presence of conjunct characters, inflexed characters, graphically similar (look alike) characters, and so on.

Apart from the aforementioned limitations, the following are the issues to be addressed while designing virtual keyboards in Indian languages.

3.4. Complexity in Design

Indian languages contain many more characters than English. Table 3 shows the sets of characters and their

TABLE 1
Summary of English Language-Based Virtual Keyboards for Text Entry in Indian Languages

Keyboard	Layout	Design Principle	Words per Minute
Dvorak (Dvorak et al., 1936)	QWERTY fashion	1. Place high-frequency characters along the middle row and depending on the right- or left-handed user. 2. More frequent characters are placed right or left of the center in middle row.	38.7 (4.75)
FITALY (Fitaly, 2009)	Single-handed layout with support for left-handed, right-handed, and mouse-driven tasks	Measure frequency of letter-to-letter transition and rearranging of the keyboard layout	41.96 (5.13)
OPTI (MacKenzie & Zhang, 1999a)	Character and bigram frequency-based layout	1. Positioning the 10 most frequent letters in the center of the keyboard. 2. Then assign 10 most frequent digraphs to the top 10 keys. 3. All the character placements are need to be done by trial and error.	42.16 (5.41)
Cirrin (Mankoff & Abowd, 1998)	Design for pen-based text entry system with the layout, which leads to cost one pen-stroke to compose a word	1. Alphabets may be arranged inside the perimeter of annulus. 2. Arrangement must be circular. 3. Place the alphabets in order to minimize the distance between likely consecutive characters.	20 (4.25)
Lewis (Lewis et al.,1999)	Alphabetic fashion	Place the characters in alphabetically in a 5 × 6 matrix	34.65 (3.87)
Hooke (Zhai et al., 2002)	Greedy layout arrangement to reduce the distance between more likely character pairs	Hooke’s law (Zhai et al., 2000)	41.15 (3.71)
Chubon (Chubon & Hester, 1988)	Follow QWERTY layout approach	QWERTY keyboard design principle	33.3 (4.06)
Metropolis (Zhai et al., 2000)	Optimum arrangement with Fitts’ digraph model	Metropolis algorithm (Zhai et al., 2000)	41.67 (4.67)
ATOMIK (Zhai et al., 2002)	Metropolis for mobile devices	Design to fulfill efficiency, alphabetical ordering, and connectivity index	41.67 (3.98)

classification in three Indian languages—Hindi, Bengali, and Telugu. Note that the table contains only base characters. In fact, in each language, many more characters (called complex and inflexed characters) can be composed with the help of these base characters. There are approximately 12 vowel characters, 33 consonants, 13 inflexions, and 10 numeric characters present in the Hindi language (Unicode Consortium, 2008b). For Bengali, 12 vowels, 35 consonants, 11 inflexions, and 10 numeric characters are used in written text (Unicode Consortium, 2008a). On the other hand, there are approximately 15 vowels, 35 consonants, 14 inflexions, and 10 numeric characters in Telugu (Unicode Consortium, 2008c).

Composing inflexed characters requires the user to press two character buttons, base character and inflexion with respect to that base. Further, composing such complex characters requires more keystrokes and eye and mouse movements. To avoid this, one strategy may be to keep them in the keyboard as units. Because the number of such units is enormously large, this eventually leads to a complex design of the keyboard, which again is not suitable to display constrained systems.

TABLE 2
Summary of Virtual Keyboards in Indian Languages

Keyboard	Layout	Design Principle	Words per minute
KeyLekh (Joshi et al., 2004)	Keyboard based on the structure of the Indic alphabet	The consonant and vowel blocks are separated and color coded to highlight the differences. The rows of keys are staggered, to allow for a typist skilled with typing in QWERTY to type comfortably.	5.09
Akshar (Mistry & Nayak, 2007)	Keyboard based on phonetic structure of the Indic scripts	Phonetic grouping system of five consonants in a group. Then assign each of these groups to a single number key on the keypad.	5.45
Microsoft (Microsoft, 2009)	QWERTY mapped	Different set of characters displayed with respect to *shift* key. All the characters are mapped with respect to hardware keyboard equivalence.	4.9
Google (Google, 2009)	Alphabetic fashion	Vowel and consonant characters are presented. On mouse hovering of consonant characters, inflexions attached with consonant form is displayed.	5.3
iLeap (Technology Development for Indian Languages, 2009)	Keyboard with phonetic support	Support several design principles, user can choose any of them at runtime.	3.7
gate2home (gate2home, 2010)	Inscript keyboard with phonetic support	Hardware mapped keyboard with support of inscript, phonetic design principles in Indian and foreign languages.	3.8
LooKeys (LooKeys, 2009)	Keyboard providing all characters and lingual combinations (dynamically changed) in Indian languages	QWERTY mapped keyboard with support of inscript, phonetic design principles in Indian and foreign languages.	4.2
Lipik (Lipik, 2007)	Hardware mapped inscript keyboard layout	Two characters are placed with respect to one key. Switching can be done with the help of shift key.	4.5
Guruji (Guruji, 2009)	Alphabetic fashion	Keys in keyboard are placed in alphabetic fashion with color grouping. Inflexed characters are present in the keyboard.	4.3
MyLanguage (MyLanguage, 2008)	Virtual keyboard with Inscript character map	QWERTY mapped keyboard with Inscript support, can be typed from hardware keyboard in particular text area.	4.4

3.5. Larger Layout

A large number of characters and inflexion symbols demands a large layout area. It is a challenging task for designers to accommodate a keyboard in an area constraint display.

3.6. Large Design Space Exploration

Theoretically a large number of arrangements of characters is possible for a keyboard (e.g., *n*! arrangements, if a language contains *n* alphabets). Out of these arrangements, there would be an arrangement for which a lower number of eye and mouse movements and hence optimum text entry is possible. Such an arrangement is called the optimum arrangement. Further, a number of layouts are possible with different groupings of characters. Finding an optimum arrangement, number of groupings, size of each group, and soon, for Indian languages turns out to be a nonpolynomial time solvable problem (Cardinal & Langerman, 2005).

4. OUR PROPOSED SOLUTION

We have studied existing design principles of virtual keyboards, which are mainly meant for text entry in English, and

TABLE 3
Character Sets in Hindi, Bengali, and Telugu Languages

Language	Vowel	Consonant	Diacritic	Punctuation	Numeric
Hindi	अ आ इ ई उ ऊ ऋ ए ऐ ओ औ अं अः अँ	क ख ग घ ङ च छ ज झ ञ ट ठ ड ढ ण त थ द ध न प फ ब भ म य र ल व श ष स ह	◌ा ◌ि ◌ी ◌ु ◌ू ◌ृ ◌े ◌ै ◌ो ◌ौ ◌ं ◌ः ◌्	। ॥	१ २ ३ ४ ५ ६ ७ ८ ९ ०
Bengali	অ আ ই ঈ উ ঊ ঋ এ ঐ ও ঔ	ক খ গ ঘ ঙ চ ছ জ ঝ ঞ ট ঠ ড ঢ ণ ত থ দ ধ ন প ফ ব ভ ম য র ল ব শ ষ স হ ড় ঢ় য় ং ঃ ঁ	◌া ◌ি ◌ী ◌ু ◌ূ ◌ৃ ◌ে ◌ৈ ◌ো ◌ৌ	। ॥	১ ২ ৩ ৪ ৫ ৬ ৭ ৮ ৯ ০
Telugu	అ ఆ ఇ ఈ ఉ ఊ ఋ ౠ ఎ ఏ ఐ ఒ ఓ ఔ	క ఖ గ ఘ ఙ చ ఛ జ ఝ ఞ ట ఠ డ ఢ ణ త థ ద ధ న ప ఫ బ భ మ య ర ఱ ల ళ వ శ ష స హ	◌ా ◌ి ◌ీ ◌ు ◌ూ ◌ృ ◌ౄ ◌ె ◌ే ◌ై ◌ొ ◌ో ◌ౌ ◌్	। ॥	౧ ౨ ౩ ౪ ౫ ౬ ౭ ౮ ౯ ౦

identified some principles that can be incorporated in an Indian language-based keyboard design task. Most of the English virtual keyboards use digraph probability (Bhattacharya, Basu, & Samanta, 2007) to place the characters in the layout. The same is also considered in our proposed design, with a proposal of placing the most frequently occurring space character at the center position. In addition, some special solutions that need to be incorporated in the case of Indian languages have been taken into consideration in our proposed layout. The basic steps involved in our approach are shown in Figure 1. Initially, a rectangular layout is chosen to accommodate five rows each having a capacity of holding 14 buttons. Then within this structure, three concentric zones have been planned. We process *Wikipedia* corpus for calculating characters' frequencies. We sort them according to their frequencies and place them at different zones such that the most frequent characters are in the inner zone, the next most frequent characters are in the layer surrounding the inner zone, and the less frequent characters are in the outermost zone. Next, we apply GA-based optimization for each zone individually to optimize eye and mouse movement. Finally, to accommodate inflexion characters, we propose a hovering window approach on the consonant characters of the language. In this section, we discuss our detailed approach to designing virtual keyboards in some of the Indian languages.

4.1. Optimum Layout of Keyboard

The proposed layout of a virtual keyboard, which we term *iLiPi* for text entry in some of the Indian languages, is shown in Figure 4. As the developed design is an "Indian language"–based text entry system that composes character by character to form a writing script (which means *Lipi* in a majority of the Indian languages), we chose *iLiPi*. The layout has the provision to accommodate 70 keys, with each key 35 × 35 pixel^2 (0.89 × 0.89 cm^2) and the total area of the layout as 514 × 185 pixel^2 (13.07 × 4.705 cm^2), measuring each pixel 0.2543 mm. This is a moderate size of a keyboard, chosen after a number of trials with the users. As space between characters occur frequently (after completion of each word) in the corpus, we have considered a space key double the size of the other keys in the layout. The concept of multizone virtual keyboard and arrangement of keys in each zone are discussed next.

A thorough user study reveals that users mostly focus on the middle region of the interface. Taking this into account, we can reduce the visual search time if we place the frequently occurring characters on or around the central region of the keyboard. If all frequently occurred characters can not be accommodated in the central region, then we may place them in multiple zones, that is, the most frequently occurring characters in the central zone, next most frequently occurring characters in the outer

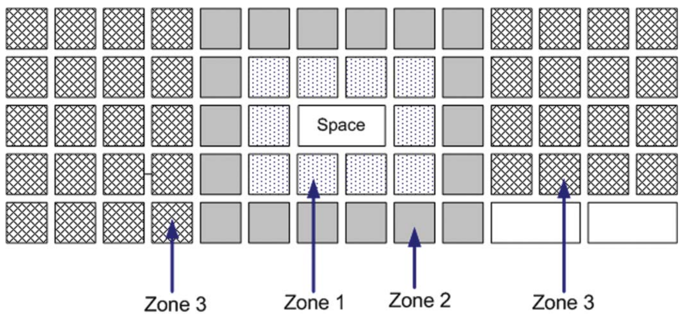


FIG. 4. Multizonal layout for iLiPi virtual keyboard (color figure available online).

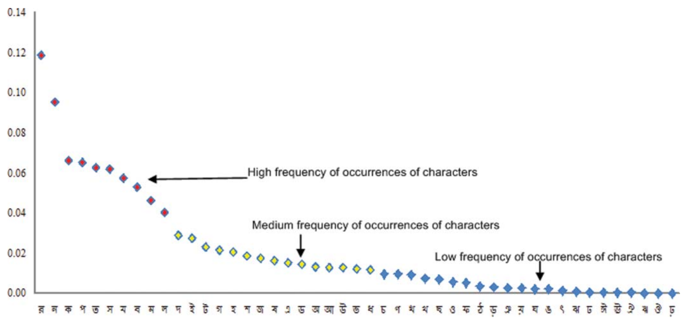


FIG. 5. Probability of occurrences of characters in Bengali language (color figure available online).

zone surrounding the center zone, and so on. Initially, we set the space character in the middle of the central zone (as it has been found that space is the highest probable character among any Indian language alphabets), then the most frequently occurring characters around it, and other less probable characters spatially arranged in other zones depending on their frequencies of occurrences (Soukoreff & MacKenzie, 1995). Note that such an arrangement is possible only if the distribution of frequency of occurrences of characters conforming to the arrangement is found for a given target language; let this be *IL*. To verify this, we must find out the frequency of occurrence of each character for the language *IL*. The same can be obtained from a (large) corpus of *IL*. From such a corpus analysis, we classify all characters in *IL* into three groups: character containing high, medium, and low frequencies of occurrences. As a case study, Figure 5 presents frequencies of occurrences of characters except space in Bengali processed from *Wikipedia* corpus of Bengali pages. Based on the distribution, we assign characters containing probability on or above 0.04013176 into a high-probable zone surrounding the space character (Zone 1) in the layout. Similarly, the characters for which probability values lie between 0.02873426 and 0.01152528 are considered to be the constituent of Zone 2. Rest of the characters are placed into a low-valued zone called Zone 3. The schematic view of such a multizonal layout is shown in Fig. 4.

Next, we intend to arrange characters in each zone in the layout so that there is a minimum mouse movement time (as between two composed words, where space is required. After adding space, there is a huge probability of getting the next word from the characters placed in the same zone around it, which results minimal mouse movement). For further fine-tuning of the key arrangement, we prioritize the positions around characters (the positions around a character are filled up with others having optimum probability as well as distance from the given character). We assume that each key contains a single character. This problem thus turns out to be an optimization problem, that is, searching an arrangement of characters on keys with respect to minimal mouse movements. We apply a genetic algorithm-based method to find an optimal arrangement of characters on keys. Our approach to genetic algorithm-based solution is described next.

The challenge is to find an arrangement of characters in three zones in order to enable minimum mouse movement time while composing texts. In this optimization problem, we consider the average mouse movement times required to compose any text as the objective function. To calculate the mouse movement time, we use Fitts' law (Fitts, 1954).

Fitts' law is defined in Equation 1.

$$MT_{ij} = a + b\log_2\left(\frac{D_{ij}}{W_j} + 1\right), \quad (1)$$

where MT_{ij} is the movement time from an i-th key to the j-th key, and a and b are two constants. The values of a and b are chosen empirically as 0.2 and 0.1, respectively (MacKenzie & Zhang, 1999a). D_{ij} denotes the Euclidean distance between the two keys i and j, and W_j is the size of the j-th key with respect to the i-th key. The value of the mean movement time ($\overline{MT}$) can be calculated using Fitts'-digraph model (Soukoreff & MacKenzie, 1995) as shown in Equation 2.

$$\overline{MT} = \sum_{i=1}^{N}\sum_{j=1}^{N} MT_{ij} \times P_{ij}. \quad (2)$$

Here, P_{ij} is the digraph probability of occurrence of j-th character after i-th character, and N denotes the total number of characters in the keyboard. The P_{ij} values for any two pair of characters can be calculated from the corpus of the language *IL*.

We follow the real coded ordered GA (Wright, 1991) to solve the aforementioned problem. We define a chromosome as three sets of characters present in three zones. The chromosome can be defined as

$$chromosome = [C_{Z_1}|C_{Z_2}|C_{Z_3}]$$

where C_{Z_1}, C_{Z_2}, and C_{Z_3} represent the sets of characters in Zone 1, 2, and 3, respectively.

We decide three sets of characters in three zones based on their frequencies of occurrences as discussed previously in this section. Note that the length of the chromosome is same as the total number of characters in *IL*. To decide an initial candidate, we choose a random arrangement of characters in their respective zones. We decide 50 such random arrangements for each candidate and consider as the initial population in our work.

After generating the initial population, we calculate the cost values based on the objective function (Equation. 2) for each candidate and arrange them in ascending order. We consider rank-based selection method (Obitko, 1998), that is, we choose 50% of the current population with higher rank values for mating pools to generate offspring in the next generation. We divide the candidate parents into two groups: One group contains 13 individuals with higher cost values, and the rest are in another group. We randomly take two parents from the two different groups and perform mating between them.

We follow the *substring* crossover technique (Hong, Wang, Lin, & Lee, 2001), which prefers that a part of the first parent should be copied in the offspring and the rest should be taken in the same order as they appear in the second parent. To be more specific, let us consider any two chromosomes *A* and *B* corresponding to two candidate parents in the current population. We consider P_1, P_2, P_3, and P_4 as the crossover points in the parts of two chromosomes (corresponding to a zone of characters). Our crossover mechanism is to copy a part of the length $|P_2 - P_1|$ from chromosome *A* and paste it into the child chromosome *C* in between P_3 and P_4 both inclusive. For the rest of the parts in chromosome *C*, we fill with characters in the order in which they appear in the parent chromosome *B*. We illustrate the proposed crossover in Figure. 6. We may note that positions P_1, P_2, P_3, and P_4 are to be chosen randomly such that $|P_2 - P_1| = |P_4 - P_3|$. The same would be repeated for all three parts to complete the crossover for an offspring.

Repeating this procedure, we construct offsprings of strength 50, the population size we have decided in each iteration. Next, we consider a mutation operation that consists of swapping the locations of two characters with respect to a zone. We have carried out *T* number of mutations at random (*T* is a number between 1 and 10 selected randomly) in the resulting chromosomes.

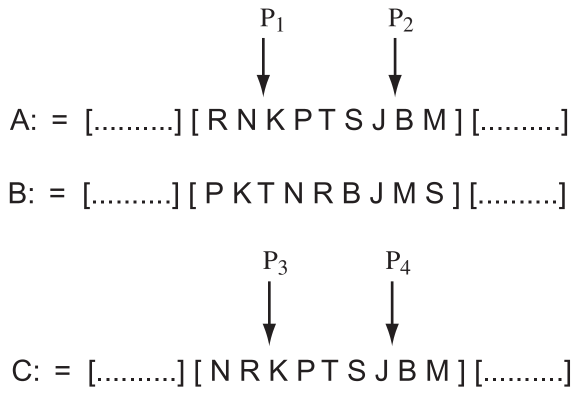

P1
P2
A: = [..........] [R N K P T S J B M] [..........]
B: = [..........] [P K T N R B J M S] [..........]
P3
P4
C: = [..........] [N R K P T S J B M] [..........]

FIG. 6. Crossover operation.

After the mutation operation is performed, we check whether the convergence is met or not. The convergence criterion is that $\overline{MT}$ difference between two individuals should be less than 0.01. If the convergence criterion is not satisfied, we move to the next generation. Otherwise, candidate corresponding to the highest rank value is selected as a solution.

4.2. Inflexion Window

In the optimal arrangement of characters in our proposed multizonal layout, we consider the arrangement of base characters in *IL*, that is, independent vowels and consonants only. Indian languages allow a large number of inflexed characters, which can be composed with consonants and inflexion symbols ("Languages of India," n.d.). Occurrences of such inflexed characters, in fact, are very frequent. In addition, there are space constraints in positioning a separate inflexion panel in the layout. Addressing this limitation, our proposal is to hover the inflexion window on a single click over the characters (only a consonant, as it only holds inflexions) to compose inflexed characters. Our approach can be explained with an example. Let us consider a part of the Hindi virtual keyboard interface as shown in Figure 7a. While a user brings the mouse pointer to a key, say च (c), the inflexion window appears (see Figure 7b) automatically, which contains all the inflexed characters possible with the base character च (c) like चा (ca), चि (ci), चे (ce), चौ (cou), and so on. After the appearance of the inflexion window, the user can tap the base character or an inflexed character, depending on which she/he wants to compose a single character or a character with inflexion.

The inflexion window can be placed at any location on the keyboard. We place the window judiciously hovering on the character itself to minimize the mouse movement while typing. As an example, suppose we place the inflexion symbols in a separate panel outside the main design area of the keyboard. Figure 8 shows that this placement uses more mouse movement than our hovering concept while composing inflexed characters. Figure 8a depicts the distance between the middlemost character of the keyboard and the rightmost inflexion symbol placed in the separate window. On the other hand, the situation explained in Figure 8b clearly shows that the hovering of the inflexion panel greatly reduces the distance between two characters.

4.3. Composing Complex Characters

Ample occurrence of inflexed and complex characters makes the text composition scenario in Indian languages more critical (MacKenzie & Tanaka-Ishii, 2007). In support of this statement, literature clarifies that 1,296 possible symbols can evolve by collating the 36 fundamental symbols of Devanagari ("Devanagari," n.d.) in which 200 may be used (176 biconsonantal [combinations of two consonants] and 24 triconsonantal [combination of three consonants] complex characters, Mukherji & Rege, 2009). For example, after analyzing Bengali

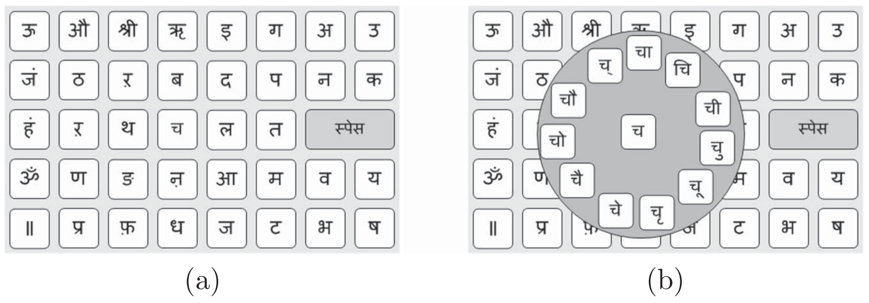


FIG. 7. Hindi virtual keyboard with and without inflexion window. (a) Hindi virtual keyboard (a part). (b) Keyboard with inflexion window.

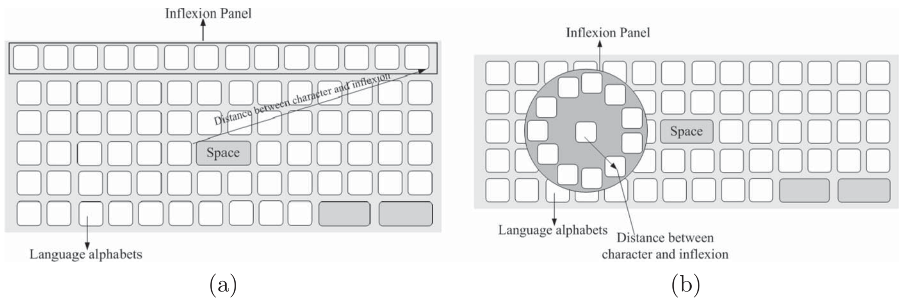


FIG. 8. Virtual keyboards with different placements of inflexion panel. (a) Virtual keyboard with separate inflexion panel. (b) Virtual keyboard with dynamic inflexion window.

language corpus, it was observed that it has 135 valid complex characters present in the language (SNLTR, 2009). To help users compose complex characters with virtual keyboards, one simple approach could be accommodating all possible combinations of characters within the keyboard itself. But due to the large number of characters, we cannot place them in the composition interface. So our objective is to provide an effective interface through which users can easily type complex characters.

In any Indian language, complex characters of different kinds can be composed by pressing three or more characters. To form a complex character, a special symbol halant/hasant (্) has been planned in Unicode-based diacritics. In other words, the special symbol ্ facilitates combining biconsonantal and triconsonantal characters. Composing complex characters in Indian languages with our proposed approach is presented next.

Units with two, three, or more consonants. A biconsonantal unit with two consonants, say C_1 and C_2, can be formed with three keystrokes in the order $C_1 +$ ্ $+ C_2$. Similarly, a triconsonantal unit with three consonants, say C_1, C_2 and C_3, can be composed as $C_1 +$ ্ $+ C_2 +$ ্ $+ C_3$. For example the word ব্রহ্ম (brohmo), where the unit হ্ম (HMo) can be composed by selecting two consonants and a hasant (্ (.h)) between them (হ (H) + ্ (.h) + ম (m)). In another case, a complex character unit ক্ষ্ণ (KShno) containing three consonants and two hasants can be composed as ক (k) + ্ (.h)+ ষ (Sh) + ্ (.h) + ণ (N). Also, we can compose characters like ক (k) + ্ (.h) = ক্ (k.h), ক্ (k.h) + ষ (Sh) = ক্ষ (kSh). On the other hand, we can consider formation of complex characters augmented with inflexions, like বিষ্ণু (biShnu), where composition needs more selection of characters (ব (b) + ি (i) + ষ (Sh) + ্ (.h) + ণ (N) + ্ (.h) + ু (u)). Similar types of construction can be achieved in Hindi and Telugu languages.

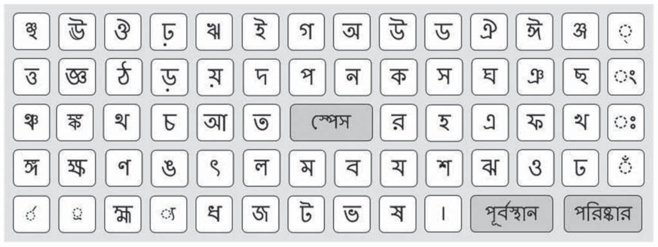
Units with special signs and symbols. Apart from biconsonantal and triconsonantal character composition, characters can be formed with the help of various signs and symbols. For example, in the Bengali language, several such signs exist (like ঁ ং ঃ র্ ্য etc.) that can be added with the base characters to form complex characters. Similarly, in Hindi, a variety of special symbols and signs are used in forming complex characters in Hindi language, which can be easily formed through our proposed virtual keyboard interface. For example, forming the word माँ (ma) (मा (ma) + ँ (chandrabindu)) requires the selection of ँ (chandrabindu) after composing the inflexed word मा (ma)), the character ज़ (ज (j) augmented with *nukta* (़) (U09DC) (GIST, 2008)). Thus, the word हिंदी (Hindi) can be composed as हि (hi) + ं (bindi) + दी (di), that is, selecting हि (hi) followed by ं (bindi) symbol from the keyboard, and so on. A similar approach is proposed for compose complex characters in Bengali and Telugu languages.

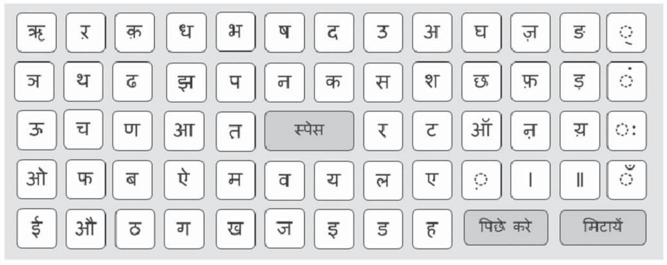
5. EXPERIMENTS, EXPERIMENTAL RESULTS, AND DISCUSSIONS

To validate the improvement and to prove the effectiveness of the proposed approach, we have conducted a number of experiments with three Indian languages—Hindi, Bengali, and Telugu. In this section we discuss our experiments and the results obtained.

5.1. Virtual Keyboards in Indian Languages

Based on the approach proposed in section 4, we have designed three keyboards—*iLiPi-B*, *iLiPi-H* and *iLiPi-T*—for text composition in Bengali, Hindi, and Telugu languages, respectively. The layouts of these three keyboards are shown in Figure 9. The Bengali virtual keyboard (*iLiPi-B*) consists of 12 vowels, 35 consonants, and 11 inflexions as shown in Figure 9a. The virtual keyboard in Hindi (*iLiPi-H*) contains 12 vowels, 33 consonants, and 11 inflexions (Figure 9b). The *iLiPi-T* virtual keyboard in Telugu consists of 15 vowels, 33 consonants, and 14 inflexions (Figure 9c). The optimal arrangements of all characters in these keyboards have been made using the approach as discussed in section 4.


(a)


(b)

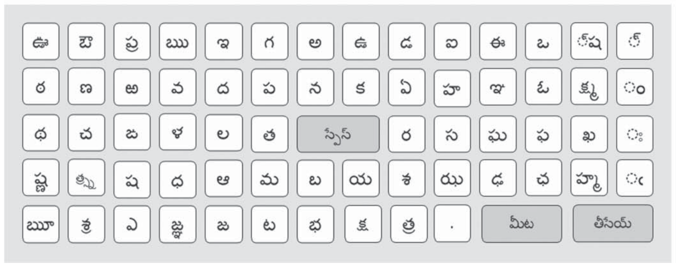

(c)

FIG. 9. Proposed iLiPi virtual keyboards in Bengali, Hindi, and Telugu. (a) Bengali keyboard: iLiPi-B. (b) Hindi keyboard: iLiPi-H. (c) Telugu keyboard: iLiPi-T.

5.2. Text Composition in Experiments

A user can compose text with any of the following methods.

- *Read and type* (read a part of text and then type the text)
- *Listen and type* (listen to a part uttered by the assistant and type the text)
- *Type freely* (type text of user's own)

In *Read and type*–based evaluation method, composition of text requires extra effort of shifting focus from reading to typing and back, which is an addition to the eye and mouse movement times. In *Listen and type*–based evaluation, users first need to listen the text word by word as uttered by the instructor and then type those words. This method inducts many errors such as phonetic, transcription, and sleeping errors (Grudin, 1983; Wobbrock & Myers, 2006). On the other hand, in the free text composition (*Type freely*) method, users are asked to type their own text, which does not require shifting focus or listening effort and is less erroneous. We follow the third strategy in our user evaluation. We consider the *form fill in* method (Shneiderman & Plaisant, 2004) and ask our user to fill a prescribed form. The form we used in our experiment consists of 25 entries (name, address, occupations, hobbies, etc.) and the need to type approximately 94 words and 480 characters.

After analyzing the collected user evaluation results from the log file, we saw that users did not select some characters in the language, as those characters did not appear in their typed texts. That problem hinders us in deciding the text entry rate in a optimal way. To overcome the suboptimality of the result, we also followed the Read and type method in which texts are decided so that all language characters present in the texts with uniform probability. Here, we consider a number of texts for users performing the Read and type method. We carefully chose the texts from existing literature in three languages in such a way that each contains huge word variation of inflexed and complex characters. We also took care that texts included almost all characters in their corresponding languages. For each language, we selected three different texts. The characteristics of the selected texts are shown in Table 4.

5.3. Participants

Fortyeight individuals participated in our experiments. Out of them, 18, 15, and 15 participants are for testing virtual keyboards in Bengali, Hindi, and Telugu language, respectively. Details of the participants are summarized in Table 5. In our experiments, we chose participants with different educational backgrounds and computer proficiency. Their educational backgrounds varied from secondary level to postgraduate level education. Thus, all participants were language literate. Participants' computer proficiency was also within a wide range, which can be classified at three different levels: (a) at

TABLE 4
Text Selection

Language	Text Under Test	Reference Text	No. of Words	No. of Complex Characters	No. of Inflexed Characters
Bengali	B_1	*Kapalkundala* by Bankim Chandra	153	23	304
	B_2	*Ramayana* in Bengali	163	53	352
	B_3	*Ashamanjababur Kukur* by Satyajit Ray	171	25	327
Hindi	H_1	*Shatranj Ki Khiladi* by Munsi Premchand	175	27	325
	H_2	*Nadi Bahati Thi* by Rajkamal Chaudhary	162	43	319
	H_3	*Prabandha-Parichaya* by Suryakant Tripathi Nirala	168	38	326
Telugu	T_1	*Bhagavatha Purana* by Bammera Pothana	103	36	313
	T_2	*Vishwambhara* by C. Narayanareddy	111	29	341
	T_3	*Kanyaasulkamu* by Gurajada Apparao	108	45	337

TABLE 5
Description of Participants

Participant's profile				No. of Participants		
Occupation	Age Group	Education	Computer Proficiency	Bengali	Hindi	Telugu
Student	19–27	Undergraduate	Level 1	2	2	1
		Higher secondary	Level 2	1	1	2
		secondary	Level 2	1	2	2
Office staff member	31–48	Graduate	Level 1	2	2	1
		Higher secondary	Level 2	1	1	1
Business person	34–53	Postgraduate	Level 1	1	2	1
		Graduate	Level 2	1	2	2
		Higher secondary	Level 3	1	1	1
Aged person	57–66	Graduate	Level 2	1	1	1
		Undergraduate	Level 3	1	1	1
Housewife	35–45	Graduate	Level 2	1	1	1
		Secondary	Level 3	2	2	1

least 5 years of experience with a computer; knowing operating systems, programming, and application software (Level 1); (b) 1 to 3 year's experience and mainly familiar with application software such as Microsoft Word, e-mail, and Internet browsing (Level 2); and (c) less than 1 year of experience and poor familiarity with computer-related tasks such as document preparation, e-mail messaging, and so on. (Level 3). We place such a variety of participants into two categories: *experienced* and *inexperienced*. This categorization is based on both educational qualification and computer proficiency level. In other words, participants with Level 1 computer proficiency are in the *experienced* category, whereas those with Level 3 proficiency are in the *inexperienced* category. Further, a participant with Level 2 computer proficiency and higher educational background (above higher secondary) belongs to the *experienced* category, otherwise, she or he is in the *inexperienced* category. There were 15 female and 33 male participants, and the average age was 29.63 years ($SD = 4.82$). Of these 48 participants, 21 participants are familiar with typing text with a mobile keypad, virtual keyboards in English or Indic languages. They belong to the experienced category. The rest of the participants are the inexperienced; they are not familiar with any text entry system and new to composition of text in any language.

5.4. Apparatus

All experiments have been conducted using 2.0GHz to 2.6GHz Pentium Core2Duo machine with HP w17e 17-in. screen LCD color monitor. The screen resolution is fixed to 1440×900. The developed keyboard interfaces for the experiments are written in C# using Visual Studio 2008, which can be accessed through mouse or single pointer-based touch screen input. The key press events are recorded automatically and stored in a log file. For tracking the mouse positions, we develop a separate window hook program written in C#. All experiments are done either in the microsoft Windows XP or in Windows 7 environment.

5.5. Evaluation Procedure

The evaluation of the proposed layouts in three Indian languages is then carried out with the experimental setup just described. There are two ways through which our evaluation process has been carried out: *user evaluation* and *model-based evaluation* (Kieras, 2002). The two evaluation procedures, parameters and metrics followed, are stated in the following.

User evaluation. In the user evaluation method, we consider two ways for typing text, namely, Read and type and Type freely. Altogether, nine benchmark texts have been selected from popular literature in the three languages for the test. We selected one short story, one novel, and one translated epic text in Bengali; one novel, one prose work, and one story in Hindi; and one novel and two stories in Telugu (also see Table 4). These testing texts are not a subset of training *Wikipedia* corpus. We conduct Read and type–based user trials in two ways. In the first way, we ask users to type as quickly and accurately as possible the randomly selected printed text supplied to them. In the other way, texts to be composed are displayed to participants on top of the text composition area in the interface. The aforementioned two ways of sessions are chosen randomly for a user. During experiments, the distance of the computer screen from the participant remains around 65 cm. The testing corpus (Table 4) and the keyboard for typing are chosen randomly in each session to minimize the memorizing effect of users. Each study lasts approximately 45 to 60 min. On average, a user spends time performing one to four experiments per day. Most of the users have performed text entry with all the keyboards and keyboard use conditions in their mother languages; only a few have not completed all the experiments successfully. A user, who has evaluated all keyboards in different conditions, took approximately two to six months to finish. Such a long duration is required to evaluate 60 (10 keyboards $\times 2$ ways $\times 3$ texts) trials per user. For both Read and type and Type freely methods, a session gap has been followed that is between 1 and 3 days. Random ordering has been followed for selecting a keyboard in user evaluation. Each user session is automatically logged into a file. After completion of a session, the session log file is analyzed to calculate text entry rate.

We also follow the Form fill in strategy to calculate text entry rate through user evaluation. In this method, we prepare three forms—F_{H_1}, F_{H_2}, F_{H_3} in Hindi; F_{B_1}, F_{B_2}, F_{B_3} in Bengali; and F_{T_1}, F_{T_2}, F_{T_3} in Telugu, where the user has to fill in personal, professional, and other details of their own. Abbreviations are not allowed in the form fields (e.g., home or workplace address). We also instruct users to fill in the fields with a nominal number of characters (e.g., while typing address of correspondence field, users should finish typing within 20–25 words, etc.). Same as in the previous experiment, in Form fill in method (Shneiderman & Plaisant, 2004), we record all activities of an entire session in a file and then analyze them.

Let T be the final transcribed or typed string entered by the user and $|T|$ be the length of this string that is the number of characters entered. Let S denote the time taken by a user in seconds, measured from the entry of the first character to the entry of the last, including backspaces. We define the text entry rate WPM (MacKenzie & Tanaka-Ishii, 2007) as shown in Equation 3. Here, $\overline{w}$ denotes average length of words for a language. It has been estimated that average word lengths are 5.11 in Bengali, 4.695 in Hindi, and 7.76 in Telugu language (Bharati, Rao, Sangal, & Bendre, 2002).

$$WPM = \frac{|T| - 1}{S} \times \frac{60}{\overline{w}} \tag{3}$$

The -1 in the numerator in Equation 3 signifies that the time S is measured from the entry of the first character to the entry of the last character (i.e., the total text length $|T| - 1$; MacKenzie & Tanaka-Ishii, 2007).

As error correction is not fully functional, for the Read and type–based text composition, each user transcribed word is checked with the word present in the corpus, and number of errors committed by the user is calculated. At the same time, a modified procedure is followed to understand different categories of errors in Type freely–based text entry. After users independently composed texts through this form-filling method, the composed or transcribed texts are carefully verified by developers in the presence of the user (as users can write nouns in terms of names of places, persons, etc., where they think that the spelling of the inputted word[s] is correct, which is not likely) to find out the errors. Further, the inexperienced people may commit mistakes in the spellings of composed words. The scenario, specifically for Form fill-in method-based text composition, can be tackled by a mechanism where, first, a hard copy of the form is to be given to users for completion, and after completion the form is digitized and users are permitted to fill in the same form using a virtual keyboard. Thus, after this session, we are able to measure the similarity between the characters written as well as typed (in this case, we consider the written text as standard). Despite being more time consuming, the procedure seems to be effective in catering to the anomalies in users' own transcribed text. Moreover, as Indian language texts consist of complex (*yuktakshar*) or inflexed characters occurring frequently, and as users commit mistakes most of the time to compose the aforementioned characters, character-level error analysis is inevitable. To compare the procedure and extend the error analysis parameters of English, after completion of each user's transcription session, a preprocessing of the composed texts is needed, where every simple and compound word can be broken down into atomic character level (like for গল্প [golpo] in Bengali language, it is to be broken into constituent character, halant, and special symbols, which needs to be tapped in on the Bengali virtual keyboard, i.e., গ[ga] + ল [la] + ্ [.h] + প [pa]. For error analysis, we apply *Minimum String Distance* (MSD) and *Key Stroke Per Character* (KSPC) metrics (Soukoreff & MacKenzie, 2003). This was decided so that we analyze errors for user evaluation but not for system simulation, as at the time of text composing, user may commit mistakes but the system cannot.

User-composed input stream through the virtual keyboard can be considered transcribed text. The input stream consists of keystrokes (some correct, some erroneous) and editing strokes (backspace, delete, cursor movements, etc.) from the keyboard. These keystrokes of the input stream can be divided into four classes, depending on their influence on the error rate. Correct keystrokes can be stated as the *Correct* (C). Errors that are not noticed by the user during composition remain in the transcribed text, which can be termed as *Incorrect and Not Fixed* (INF). Together, these C and INF keystrokes comprise all of the characters in the transcribed text. The errors corrected characters are grouped in the *Incorrect but Fixed* (IF) category. Similarly, the keystrokes performing the corrections are considered as *Fix* (F). Given the presented text, input stream, and transcribed text, it is easy to classify keystrokes.

C and *INF*: These represent the characters in the transcribed text that belong to the corrected or not corrected classes, respectively. The *C* keystrokes are the correct characters in the transcribed text, and *INF* contain wrongly entered keystrokes present in final transcribed stream.

F: The keystrokes belonging to the *F* class are mainly editing functions like backspace, delete, and cursor movement as well as modifier keys (Shift, Alt, and Control).

IF: The *IF* keystrokes are those that are in the input stream but not present in the transcribed text. Also, editing keys are exempted from this class.

For classifying the particular characters in each group, a more detailed character-by-character analysis of errors is required (Soukoreff & MacKenzie, 2002). However, in computing error rates, the particular characters are not useful in calculation. Rather, the size of the classes is important. Therefore, for convenience, let C, INF, IF, and F denote the number of keystrokes in each of their respective classes. The MSD error rate and KSPC statistic then can be defined in Equations 4 and 5, respectively (Soukoreff & MacKenzie, 2003).

$$MSDErrorRate = \frac{INF}{C + INF} \times 100\% \qquad (4)$$

$$KSPC \quad \frac{C + INF + IF + F}{C + INF} \qquad (5)$$

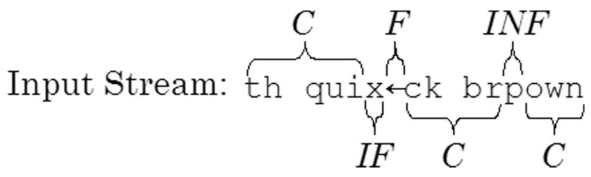


FIG. 10. Classifying the keystrokes in an example.

Consider the following example for further classification of the classes just stated.

Presented text: the quick brown
Input stream: th quix←ck brpown
Transcribed text: th quick brpown

In this example, there are three errors: An "e" is omitted. There is an extra "x," which is corrected with a backspace. Also, there is an extra "p" that remains uncorrected. The keystrokes are mapped into the several class depicted in Figure 10.

The example, shown in Figure 10, contains the following information: the INF class also includes the missing "e" keystroke, which is incorrect. In this example, C = 14, INF = 2 (counting the extra "p,", and the missing "e"), IF = 1, and F = 1.

Model-based evaluation. We use the keystroke level model (KLM; Kieras, 1993) along with Fitts' digraph model (Soukoreff & MacKenzie, 1995) in our model-based evaluation. The parameters and metrics considered in our model-based evaluation are summarized as follows. The KLM is used to evaluate a user interface corresponding to the task completion time so far as the lower level tasks are concerned. In addition, the KLM also takes care of cognitive load. The different lower level tasks, cognitive load, and their completion times according to the KLM are summarized in Table 6. So far as the evaluation of virtual keyboard interface is concerned, the KLM can be applied to evaluate mouse-related task completion time plus the cognitive load. However, there is a modification that needs to be performed to evaluate mouse movement time properly in its present scope. This time would account for the time required for mouse movement from a currently selected key to the next key. To measure this component, we propose to use Fitts' digraph model (Soukoreff & MacKenzie, 1995), which is already discussed in Section 4 (see Equation 2).

Thus, the time for typing a single character $T_{compose}$ can be evaluated by measuring the sum of the times taken to perform three important subtasks individually: mentally preparing for tapping next character (T_M) (which involves searching for the character an the keyboard by visual perception through the eyes and finding the decision through cognition in the human brain), moving or pointing (T_P) the mouse pointer toward the newly decided characters ($\overline{MT}$) calculated using Fitts'digraph model, and finally clicking on the character button by mouse (T_{BB}). In the KLM model, it has been pointed out that the system

TABLE 6
Parameters in KLM

Operator Type	Operator	Remarks	Time (in seconds)
Physical motor operator	B BB	Pressing or releasing mouse button Clicking the mouse button	0.10 (down or up) 0.20 (click)
Mental operator	M	Mentally preparing for a physical action	1.35
System response operator	R	Waiting for system to respond	0.1

response operator (T_R) value depends on the performance of the system where the model is to be deployed. In our case, we take T_R as 0.1 5 (Kieras, 1993). The tasks with their predicted values are mentioned in Table 6.

$$T_{compose} = T_{BB} + T_M + T_R + T_P(\overline{MT}) \qquad (6)$$

In the calculation of $\overline{MT}$ (Equation. 2), the values of a and b are decided empirically as 0.2 and 0.1, respectively (MacKenzie & Zhang, 1999a). Width W_j of all target keys are taken as 1 except Space, Clear, and Back. For these three keys we take W_j as 2. The digraph probability P_{ij}'s are calculated following the Wikipedia corpus of target *IL*. Thus, using KLM and Fitts'digraph model, we can estimate the average time to type a character ($T_{compose}$) as stated in Equation. 7a and 7b, respectively.

$$CPS = \frac{1}{T_{compose}} \qquad (7a)$$

$$WPM = CPS \times \left(\frac{60}{\overline{w}}\right) \qquad (7b)$$

Here, $\overline{w}$ denotes that the average word length of the language *IL*.

5.6. Design

Existing virtual keyboards in Indian languages like iLeap (Technology Developmentfor Indian Languages, 2009), gate2home (gate2home, 2010), LooKeys (LooKeys, 2009), Lipik (Lipik, 2007), MyLanguage (MyLanguage, 2008), Google (Google, 2009), Guruji (Guruji, 2009), and Avro (OmicronLab, 2010) are considered for the experiment. In window-based system, Microsoft offers an on-screen keyboard in Indian languages, which is also taken into account. On the other hand, keyboards developed based on the design principles of popular English keyboards like Dvorak, OPTI, FITALY, Lewis, Hooke, Metropolis, and ATOMIK in Hindi, Bengali, and Telugu languages are also taken for experiment. Moreover, to prove the effectiveness of the proposed keyboard design, four different layouts with different design principles have been investigated, which are also accessed by users during experiments. Those alternate designs are described next.

Design 1 (Layout in alphabetical order). This design considers alphabetical ordering of characters in the layout. The keyboard consists of a distinct individual grouping of vowels, consonants, and inflexions, and characters are spatially laid in their respective groups in alphabetical order (Figure 11a).

Design 2 (Layout with random arrangement of characters). In this design, we consider a random arrangement of characters except inflexions and punctuation symbols in the layout (see Figure 11b).

Design 3 (Layout of multiple zones). In this design, we arrange the characters into three different zones. The most frequently occurring characters are placed in the central zone followed by the next most frequently occurring characters in the outer zone surrounded by the central zone, and so on (Figure 11c).

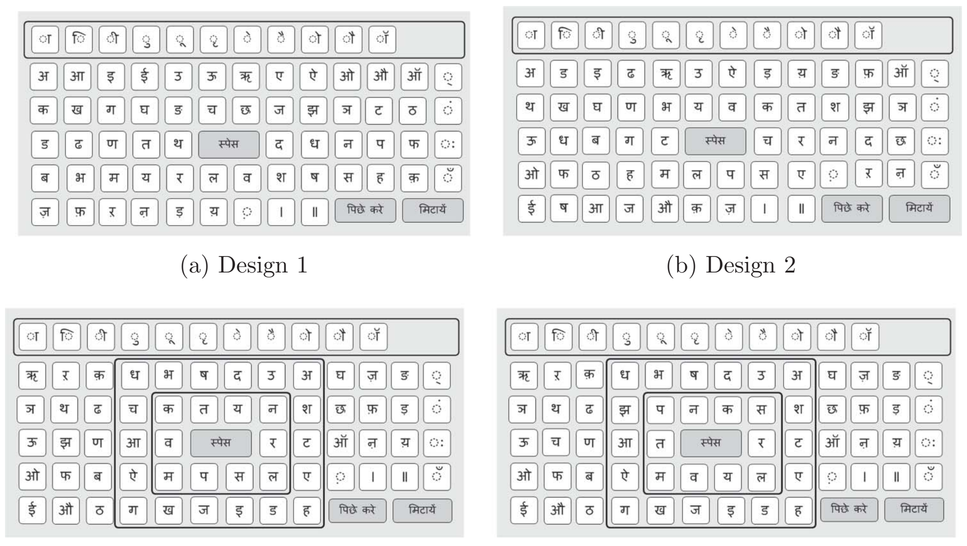
Design 4 (Multizonal with GA-based optimum key placement). This design is the same as Design 3 except that in each zone of the layout, characters are placed with an optimal arrangement. The layout of Design 4 is shown in Figure 11d.

5.6. Experimental results

The main objective of our experiments is to study the text entry rates of the existing English and Indian language virtual keyboards along with three alternate designs and proposed iLiPi keyboard through user as well as model-based evaluations. Also, we analyzed the layout area requirements for all the keyboards. The other goals are to analyze the learning curve obtained based on longitudinal user studies on several keyboards as well as to investigate errors committed by users during text composition.

Further, to analyze the results of text entry rates from text typing sessions performed by users or computer programs, we conducted an analysis of variance (ANOVA) test. The tests are conducted using *Statistical Package for the Social Sciences* (SPSS tool; IBM, n.d.). As the average word lengths in these three languages are different, the comparison may not be properly accomplished. So to avoid the ambiguity, we take the lower valued Hindi word length as a baseline and convert the other languages text entry rate into one standard by multiplying with a factor of $\frac{4.69}{x}$, where x denotes the average word length of a target language.

Also, we consider weighted mean of text entry results in three languages (using Equation 8) for statistical analysis.



(a) Design 1 (b) Design 2

(c) Design 3 (d) Design 4

FIG. 11. Alternate virtual keyboard design approaches in Hindi. (a) Design 1. (b) Design 2. (c) Design 3. (d) Design 4.

$$\textit{Weighted mean of WPM} = \frac{\sum_{i=1}^{n} [WPM_i \times \textit{Average word length}_i]}{\sum_{i=1}^{n} \textit{Average word length}_i}, \tag{8}$$

where i indicates the individual language.

To comprehend user acceptance of alternate keyboard designs, we conducted experiments with users through the Read and type–based text composition method with selected text corpus (mentioned in Table 4) in Bengali, Hindi, and Telugu languages. The measured mean text entry rates for those designs along with proposed iLiPi-H in Hindi are depicted graphically in Figure 12.

From the result we see that Design 2 gives 0.72% better text entry rate than Design 1. Similarly, Design 3 and Design 4 give 6.88% and 13.77% better text entry rates when we compare them with respect to Design 1, respectively. We note that 18.84% more text entry rate is possible with iLiPi-H compared to Design 1. Further, we note that the design concept of inflexion window (iLiPi-H) enables 4.46% better text entry rate compared to noninflexion window-based design (Design 4). Similarly, Design 4 achieves 6.44% improvement in text entry rate over Design 3, and Design 3 achieves 6.12% better text entry rate than Design 2. More or less similar results have been observed with iLiPi-B and iLiPi-T.

Further, the ANOVA on text entry speeds shows that there is a significant difference between the means of users' performance on different keyboard designs, $F_{(4,35)} = 18.92, p < .05$.

Text entry rate. We conducted both user evaluation and model-based simulation experiments. It may be noted that for each of the three languages, the inexperienced users are asked

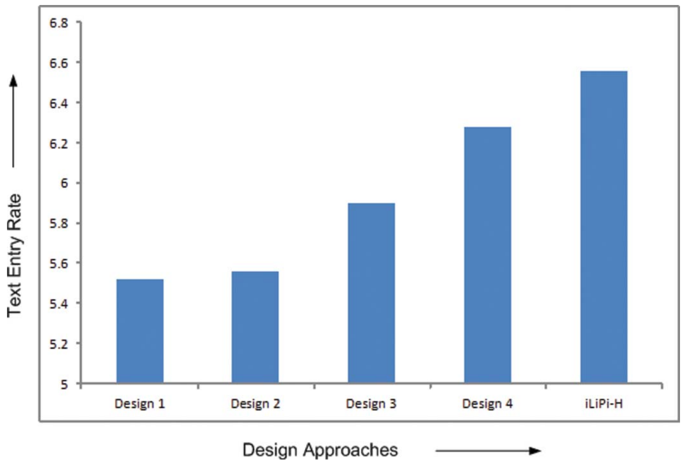


FIG. 12. Comparison among different virtual keyboard designs (color figure available online).

to spend some time (may be 1 or 2 days) composing text using virtual keyboards before starting the actual sessions to get familiarized. After gathering the result from log files, we compare the means of text entry rates taken from several sessions performed by experienced or inexperienced users as well as automated systems of the proposed design approach with respect to some existing designs. The text entry performances in comparison with different keyboard design approaches are given next.

User evaluation. Participants have tried to compose selected texts (mentioned in Table 4) in three languages (Bengali, Hindi, and Telugu) through Indian language-based existing virtual keyboards like iLeap (Technology Development for Indian Languages, 2009), gate2home (gate2home, 2010), LooKeys (LooKeys, 2009), Lipik (Lipik, 2007), MyLanguage

(MyLanguage, 2008), Google (Google, 2009), Guruji (Guruji, 2009), and Avro (OmicronLab, 2010). During the analysis of the keyboards from their original source, we found that some layouts are lacking in providing a sufficient text area attached to the keyboard, like Google. In that case, to increase the usability and overcome the current limitations, we imitated the same design and attached a moderately large text area with each keyboard. The experiments on keyboards have been performed through several sessions, and text entry rates are measured. In a window-based system, Microsoft offers an on-screen keyboard in Indian languages, which is also taken into account. The mean experimental user evaluation results on different test corpus in three Indian languages have been gathered. The mean results of both experienced and inexperienced participants using both text composition methods (Read and type and Type freely) in three different languages are shown in Table 7.

Subsequent analysis of user results using the Read and type method for typing reveals that the developed iLiPi design achieves more text entry rate than Google keyboard in Hindi, Avro keyboard in Bengali, and Guruji keyboard in Telugu by 15.41%, 18.53% and 22.12%, respectively, for inexperienced users as well as 20.70%, 18.96%, and 16.57% for experienced users. Similarly, for Type freely–based text entry, iLiPi outperforms the same keyboards in three languages by 14.95%, 17.35%, and 19.81% for inexperienced users and by 19.13%, 17.05%, and 18.90% in the case of experienced users.

To further compare the two types of user performance on different virtual keyboard-based text composition mechanisms, we conducted the ANOVA tests for Indian language and English-based virtual keyboards in three languages. While comparing the proposed iLiPi keyboard with existing Indian language keyboards, the ANOVA tests performed for experienced and inexperienced users on Read and type–based text entry reveal that the mean text entry rate of the keyboards are not the same, $F_{(9,476)} = 42.34, p < .05$. On the other side, while we compare the results obtained from users performance for composing text freely (Type freely method), we also observe that there is significant difference between performance of Indian virtual keyboards, $F_{(9,476)} = 37.96, p < .05$. The statistical test, that is, ANOVA test, has been conducted for comparing mean text entry rates of inexperienced and experienced users. The result substantiates that there is significant difference between performance of keyboards irrespective of users' proficiency—for inexperienced, $F_{(9,233)} = 33.29, p < .05$; for experienced, $F_{(9,233)} = 22.54, p < .05$.

We calculated the mean text entry rates for two kind of users on Indian language keyboards and analyzed the result through an independent T test. There is a significant difference between the mean text entry rates (WPM) achieved by experienced and inexperienced users, $t(18) = -3.12$, $p < .05$, for a two-tailed test. The result reflects that for a fixed type of text composition task and text to be typed, experienced users outperform users in the inexperienced category. Similarly, we gathered users' text composition results, keeping other constraints fixed, and analyzed the tendency of the data. The statistical test (independent T test) was conducted, which showed that there exist significant differences between the mean text entry rates achieved through the Read and type and Type freely methods, $t(18) = -2.122$, $p < .05$, for a two-tailed test. Experimental results confirm that, for other unaltered conditions, experienced users and the Type freely method proved to be better than inexperienced users and the Read and type method, respectively.

TABLE 7
User Evaluation Result for Existing Indian Language-Based Keyboards

			Text Entry Rate (in WPM)									
Language	Text entry method	Type of user	iLeap	gate2home	Lookeys	Lipik	Google	Guruji	Microsoft	Avro	MyLanguage	iLiPi
Hindi	Read & Type	Inexp	4.30	4.24	4.29	5.04	5.32	5.25	4.53	—	4.34	6.14
		Exp	4.58	4.42	4.47	5.32	5.70	5.63	4.64	—	4.52	6.88
	Type Freely	Inexp	4.96	4.72	4.83	5.28	5.55	5.48	4.89	—	4.82	6.38
		Exp	5.24	5.18	5.12	5.96	6.22	6.14	5.29	—	5.19	7.41
Bengali	Read & Type	Inexp	4.03	3.90	3.94	4.40	4.64	—	4.05	4.69	3.96	5.50
		Exp	4.49	4.36	4.39	4.93	5.38	—	4.69	5.37	4.44	6.40
	Type Freely	Inexp	4.30	4.15	4.19	4.66	4.90	—	4.31	4.91	4.16	5.75
		Exp	4.84	4.61	4.65	5.19	5.61	—	4.95	5.63	4.67	6.59
Telugu	Read & Type	Inexp	2.66	2.61	2.59	2.76	3.05	3.12	2.76	—	2.63	3.81
		Exp	3.03	2.91	2.94	3.23	3.35	3.38	3.10	—	3.03	3.94
	Type Freely	Inexp	2.82	2.76	2.74	3.13	3.19	3.23	2.80	—	2.79	3.87
		Exp	3.19	3.05	3.11	3.40	3.44	3.51	3.25	—	3.19	4.09

Note. Inexp-inexperience; Exp-experience.

The keyboards based on the design principles of Dvorak, OPTI, FITALY, Lewis, Hooke, Metropolis, and ATOMIK are adopted for virtual keyboards in Hindi, Bengali, and Telugu. We compare these designs with our designs through users' evaluation, maintaining the same condition and corpus. The mean user evaluation results for all users are shown in Table 8.

For the Read and type method-based typing, the developed iLiPi design yields higher text entry rate than the existing highest text entry valued OPTI keyboard in the Hindi, Bengali, and Telugu languages by 17.85%, 28.80%, and 29.59%, respectively, for inexperienced users and 21.34%, 33.11%, and 18.60%, respectively, for experienced users. Similar observations have been observed for the other method of composing text, namely, Type freely, in the Hindi, Bengali, and Telugu languages by 15.85%, 21.99%, and 22.08%, respectively, for inexperienced users and 19.32%, 28.72%, and 16.19% for experienced users.

We performed an ANOVA to compare the results of the proposed iLiPi keyboard with other English-language-based keyboards. The test has been performed on experienced and inexperienced users' text composition rates achieved through the Read and type and Type freely methods. The result substantiates that for the Read and type method, the mean text entry rate of 10 virtual keyboards is not the same, $\Delta F(9, 530) = 45.79, p < .05$. We analyzed the results gathered from users' performance for freely composed text and observed that there is also a significant difference between the performance of English virtual keyboards, $F(9, 530) = 22.87, p < .05$. The ANOVA test conducted to verify mean text entry rates in user base (i.e., for inexperienced and experienced users) also presents that there is a significant difference between the performance of keyboards irrespective of users' proficiency: for inexperienced, $F(9, 260) = 26.92, p < .05$; for experienced, $F(9, 260) = 24.64, p < .05$.

Also, further statistical results reveal that there are significant differences between the means of text entry rates accomplished by experienced and inexperienced users, $t(18) = -2.31, p < .05$, for a two-tailed test, as well as Read and type and Type freely–based text composition methods, $t(18) = -2.29, p < .05$, for a two-tailed test, keeping other constraints unchanged. The results inevitably support the normal circumstances.

Model-based evaluation. The performance of a computing system to compose text automatically can be measured through the KLM model. Using the model, the system's text composition rates on previously mentioned corpus through virtual keyboards in three different languages are depicted in Tables 9 and 10.

An analysis of system-performed results signifies that the iLiPi design achieves a better text entry rate than Google keyboard in Hindi (21.98%), Avro keyboard in Bengali (21.34%), and Guruji keyboard in Telugu (33.97%), respectively. The same observations have been achieved for English-language-based keyboard for the Hindi (26.22%), Bengali (28.99%), and Telugu (29.85%) languages.

Statistical tests have been conducted to scrutinize the trend of model-based evaluation results executed on different virtual keyboards. The comparison between weighted mean text entry rates carried out for existing Indian languages as well as English-language-based keyboards confirms a significant difference between them. The ANOVA test on Indian language keyboards produces significant difference, $F(9, 71) = 12.44, p < .05$, and English-language-based keyboard performance results also follow the same trend, $F(9, 80) = 15.38$, $p < .05$.

TABLE 8
User Evaluation Result for English Language-Based Keyboards in Indian Languages

Language	Text entry method	Type of user	Text entry rate (in WPM)									
			Dvorak	Fitaly	Opti	Lewis	Hooke	Cirrin	Chubon	Metropolis	Atomik	iLiPi
Hindi	Read & Type	Inexp	5.08	5.12	5.21	4.93	4.36	4.67	4.42	4.84	4.99	6.14
		Exp	5.55	5.57	5.67	5.32	4.75	4.81	4.83	5.19	5.32	6.88
	Type Freely	Inexp	5.47	5.51	5.53	5.42	4.76	5.02	4.76	5.08	5.19	6.38
		Exp	6.06	6.13	6.21	6.08	5.37	5.56	5.89	5.95	5.93	7.41
Bengali	Read & Type	Inexp	4.03	4.23	4.27	4.14	3.84	3.92	3.86	3.91	3.99	5.50
		Exp	4.36	4.38	4.41	4.32	3.99	4.34	4.03	4.08	4.16	5.87
	Type Freely	Inexp	4.06	4.25	4.32	4.17	3.88	4.01	3.87	3.96	4.06	5.27
		Exp	4.58	4.63	4.70	4.65	4.28	4.64	4.30	4.37	4.41	6.05
Telugu	Read & Type	Inexp	2.90	2.93	2.94	2.87	2.73	2.82	2.77	2.85	2.85	3.81
		Exp	3.29	3.27	3.32	3.14	3.04	3.21	3.17	3.15	3.19	3.94
	Type Freely	Inexp	3.03	3.08	3.17	3.05	2.85	3.05	2.87	3.05	3.03	3.87
		Exp	3.43	3.45	3.52	3.38	3.26	3.40	3.28	3.31	3.37	4.09

Note. Inexp = inexperience; Exp = experience.

TABLE 9
Model-Based Evaluation Result for Existing Indian Language-Based Keyboards

	Text entry rate (in WPM)									
Language	iLeap	gate2 home	Lookeys	Lipik	Google	Guruji	Microsoft	Avro	My Language	iLiPi
Hindi	5.55	5.53	5.54	5.55	5.96	5.70	5.56	–	5.53	7.27
Bengali	4.78	4.76	4.77	4.79	5.06	–	4.83	5.03	4.75	6.14
Telugu	2.75	2.78	2.77	2.74	3.15	3.08	2.78	–	2.76	4.22

TABLE 10
Model-Based Evaluation Result for English Language-Based Keyboards in Indian Languages

	Text entry rate (in WPM)									
Language	Dvorak	FITALY	OPTI	Lewis	Hooke	Cirrin	Chubon	Metro polis	ATOMIK	iLiPi
Hindi	5.72	5.74	5.76	5.53	5.26	5.66	5.22	5.35	5.23	7.27
Bengali	4.72	4.74	4.76	4.64	4.30	4.63	4.26	4.38	4.27	6.14
Telugu	3.18	3.19	3.25	3.09	3.01	3.06	2.97	2.99	3.06	4.22

TABLE 11
Layout Area of Different Keyboards (*in* cm^2)

Indian Language-Based Virtual Keyboards		Existing Keyboard Design Approaches of English Applied to Indian Languages	
Keyboards	Layout Area (in cm^2)	Keyboards	Layout Area (in cm^2)
iLeap	20.47 × 6.38	Dvorak	18.66 × 6.41
gate2home	14.98 × 5.21	FITALY	18.67 × 7.32
Lookeys	16.53 × 6.69	OPTI	19.12 × 7.43
Lipik	16.99 × 6.03	Lewis	18.36 × 5.72
Google	10.53 × 10.58	Hooke	16.43 × 5.52
Guruji	11.16 × 6.61	Cirrin	15.11 × 5.19
Microsoft	15.21 × 7.58	Chubon	15.41 × 6.18
Avro	14.04 × 5.62	Metropolis	17.24 × 5.98
My Language	13.53 × 5.24	ATOMIK	16.61 × 7.15
iLiPi	13.07 × 4.71	Single-finger	13.02 × 12.97

Layout area. The keyboard occupied area for existing Indian-language-based virtual keyboards and English-language-design principle-based keyboard layouts are shown in Table 11. Here we considered the standard screen area to be 1440 × 900 pixel^2, that is, 36 × 22.5 cm^2.

Table 11 reflects the fact that the proposed iLiPi layout occupies on average a 514 × 185 pixel^2 area (13.07 × 4.705 cm^2) on the screen (36.62 × 22.89 cm^2), and it is 15% lower than any average English-language-based layout occupancy area as well as on average 12% less than existing Indian-language-based virtual keyboards.

Learning curve and error proneness. We also gathered feedback regarding user friendliness and error proneness from the users after completion of the evaluation sessions for every keyboard and analyzed them thoroughly.

Learning curve. Due to space limitations, we plot only Bengali-language-based text entry performance results (for 60 consecutive sessions) of three virtual keyboards, each representing one particular type of design approach. Those keyboards are OPTI, Avro, and an ad hoc virtual keyboard (Design 1). We have taken the users' text composition performance results for 60 sessions spread over a

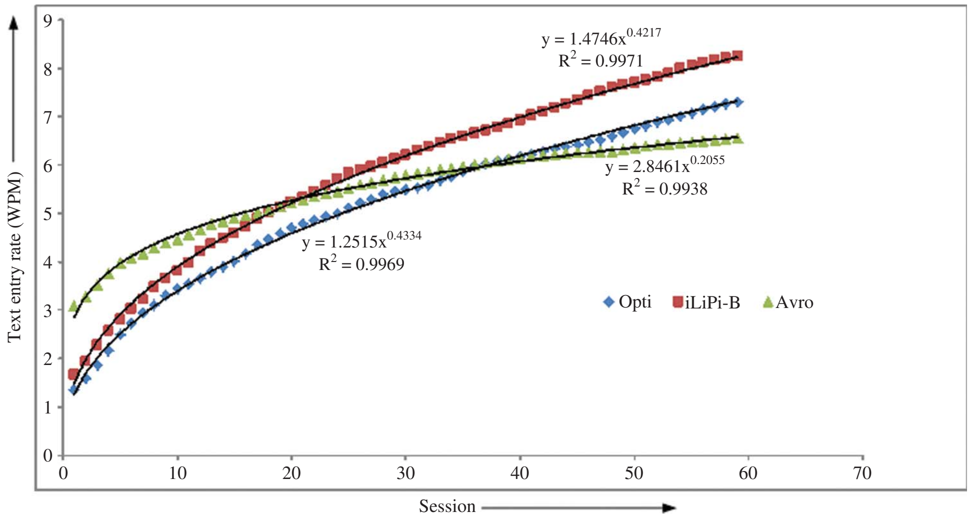


FIG. 13. Learning curve (color figure available online).

duration of 3 weeks and Type freely followed text composition strategy.

The observed results are shown in Figure 13. From the figure, it is evident that iLiPi-B takes more initial effort to learn compared to OPTI and Avro; however, after 20 to 25 sessions, iLiPi-B outperforms other two. The iLiPi-B layout reaches nearly 5.71 WPM by the 25th session, whereas the performance of the OPTI and Avro keyboard layouts achieve up to 5.23 WPM text entry speed on average.

For each layout, we derived standard regression models in the form of the power curve fitting as it follows the *Power law of learning* (Card, Moran, & Newell, 1983). The prediction equations and the squared correlation coefficients for the curves are illustrated in Figure 13. The high R^2 values signify that the learning curves are fitted more accurately with user behavior. In three cases, more than 99.3% of the variance is observed in the models. A higher R^2 value indicates that participants are becoming accustomed to computers and they are familiar with the virtual-keyboard-based text composition. The longitudinal study lasts for 60 sessions for each experienced and inexperienced user. The participants become "experienced" in the context of composing text through virtual-keyboards. The text typing performed with iLiPi-B layout lasts for several days. So, we plot the results and construct the learning curves, which inevitably reflects the increasing efficiency of users after performing several sessions (see Figure 13). The highest text entry rate achieved by a user on typing similar-length texts for the iLiPi-B layout is 8.25 WPM and for the OPTI and Avro layouts is 7.31 and 6.55 WPM, respectively.

Error proneness. We also observed the number of errors committed by users during text composition experiments. We used the MSD/KSPC metric (Soukoreff & MacKenzie, 2002; described in section V[E]) for calculating error rate and considered the method sufficient toward fulfilling the requirement estimating the approximate error rate. Error corrections during the user evaluation experiments (through Read and type and Type freely tasks) are allowed. Two kinds of errors have occurred, those that are corrected and those that are not corrected. The minimum string distance (MSD) is used to count the errors that are not corrected (*INF* error). The KSPC is used for calculating the number of keystrokes used to correct errors. If there are no extra keystrokes, KSPC is equal to 1. If some errors are corrected (*IF*), KSPC is greater than 1.

An ANOVA reveals that there is no significant difference in error rates between the keyboard designs, $F(14, 269) = 19.74, p < .05$. Another observation shows that there is a significant difference in committing error rates between experienced and inexperienced users, $t(18) = -3.27, p < .05$, for a two-tailed test.

5.7. Discussion

The concept of multizonal layout and optimal arrangement of characters in each zone proves to be an effective solution so far the reduced hand (and eye) movements and hence a better text entry rate is concerned. In fact, we achieve 14% more text entry with this design concept compared to the design with alphabetical ordering of the characters within a single zone (like, for Bengali, Avro keyboard; OmicronLab, 2010). Further, the inclusion of inflexion window yields 5% more text entry rate than the keyboard with multizonal and optimal placement of characters. In addition to this, the concept of inflexion window reduces display space overhead required for on-screen graphics keyboard. This is indeed a desirable solution to develop a text entry systems in Indian languages.

Text entry rate with the proposed design approach is also comparable with the existing virtual keyboards for text entry in Indian languages, such as Google (for Hindi language), Avro (for Bengali), Guruji (for Telugu), and so on. With compared to Google and Guruji virtual keyboard, iLiPi designs provide

17.55% and 19.35% more text entry rates in Hindi and Telugu, respectively. Similarly, iLiPi design in Bengali achieves 17.97% better text entry rate than Avro keyboard. Further, our design approach is found to be superior compared to the virtual keyboards based on design principles, which yield impressive result in terms of text entry rate, such as OPTI. Compared to OPTI, our design is 20.47%, 28.16%, 21.62% efficient, on average, for text compositions in Bengali, Hindi, and Telugu, respectively.

Although the proposed design approach demands more learning than others, it would not be an issue as experimental result substantiates that after 20 to 30 sessions, users become more friendly with pointing characters in the keyboards. However, as far as the error proneness is concerned, all designs including our design produce similar outcomes.

Threats to validity. Related to our experiments and experimental results, we would like to point out their validity and limitations, if any.

External validity. We have followed two modes of evaluation: user and model-based. For the user-based evaluation, we considered 35 to 50 users with various backgrounds (novice, intermediate, and expert). Further, we follow two strategies: *Form fill in* (where users enter their own text) and Read and type (with some carefully chosen texts). The results of the user evaluation is therefore subject to the limitations on number of subjects involved, forms considered, and texts chosen in our experiments. In a model-based evaluation, we have considered Fitts' digraph law for measuring the mouse movements and KLM to measure other activities (motor activity for button press and release, cognitive activity for deciding the next target, etc.). Using other task-oriented evaluation tool, namely, ACT-R (Anderson et al., 2004), model-based evaluation can also be performed.

Internal validity. Text entry rate measurement may be influenced by some other design parameters: layout size, sizes of keys, sizes of fonts on keys, background and foreground colors of the fonts, and so on. Hence, our experimental results are under the assumptions of the default settings of these parameters. Although these settings have been chosen after a number of trials and consultation with domain experts. It would be another interesting matter to validate the results with other platforms such as cell phones, and iPods (small display devices). In this work, our experiments are restricted to desktop environments only.

Construct validity. We consider data in user-based evaluation after a number of sessions (40 to 50 sessions in a wide time frame of 3 months), when we confirm insignificant fluctuations ($\pm 5\%$) in the observations. The optimal arrangement of character placement in the layout is governed by the factors digraph probability and genetic algorithm (GA). We have considered the *Wikipedia* corpus to calculate digraph probabilities for languages aimed in this work. It has been verified that the sizes of the corpus are comparable to the sizes of other sources such as daily newspapers. Optimization problem solving could be carried out with optimization methodology such as Simulated Annealing, Ant Colony Optimization, and Particle Swarm Optimization, as an alternative to GA. We have chosen GA because of its better fit with the objective function, faster convergence rate, and hence faster computation.

6. CONCLUSIONS

Virtual keyboards are becoming very popular for composing texts in small display devices. In recent times many virtual keyboards have been developed, which are mainly for the composition of texts in English. However, many users (18.49% of the total world populations; Statistics, 2010) are illiterate in English and need virtual keyboards in their mother languages. This work attempts to bridge this gap in the context of Indian scenarios. This work aims to design virtual keyboards in Indian languages, and more specifically the three most read and spoken Indian languages, namely, Hindi (second-most spoken and written language, because 7.539% of the world population [Vistawide, 2009] use this as a mother and second language), Bengali (seventh-most widely spoken and written language in the world as 3.308% of the world population [Vistawide, 2009] use this as a mother and second language), and Telugu (with respect to the number of speakers around the world, it is the 15th language, as 1.154% of the world population [Vistawide, 2009] use this as a mother, and second language). There are some virtual keyboards known in these languages (Google, 2009; MyLanguage, 2008; OmicronLab, 2010); nevertheless, they are based on adhoc design principles and do not necessarily provide better text entry rates. The virtual keyboards developed according to our proposed approach yield a 21% higher text entry rate compared to the best among existing keyboards. Apart from this, the proposed keyboards are more ergonomic for composing complex characters and inflexed characters with fewer keystrokes. Further, our approach provides an area-efficient solution dealing with typical language features in Indian languages and hence makes the keyboards potential candidates to be applied in text entry tasks in mobile applications. This way the proposed work would cater to growing demands not only in India but also in neighboring countries like Nepal (the majority speak and read Hindi) and Bangladesh (Bengali is the national language of this country). In fact, the proposed approach can be extended to include many other languages in India as well as in other countries.

REFERENCES

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Quin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*, 1036–1060.

Bengali script. (n.d). In *Wikipedia*. Retrieved July 2010 from http://en.wikipedia.org/wiki/Bengali_script

Bharati, A., Rao, P., Sangal, R., & Bendre, S. M. (2002). Basic statistical analasysis of corpus and cross comparision. In *Proceedings of ICON*. Hyderabad, India: Language Translation Research Centre, IIIT Hyderabad.

Bhattacharya, S., Basu, A., & Samanta, D. (2007). Performance models for virtual scanning keyboards: Reducing user involvement in the design. In *Proceedings of the International Conference on Information and Communication Technologies for Development*. Bangalore, India: ACM/IEEE.

Brown, C. M. (1988). *Human–computer interface design guidelines*. Norwood, NJ: Ablex.

Card, S. K., Moran, T. P. & Newell, A., (1983). *The psychology of human–computer interaction*. Hilldale, NJ: Erlbaum.

Cardinal, J., & Langerman, S. (2005). Designing small keyboards is hard. *Theory of Computer Science*, *332*, 405–415.

Chubon, R. A., & Hester, M. R. (1988). An enhanced standard computer keyboard system for single-finger and typing-stick typing. *Journal of Rehabilitation Research and Development*, *25*(4), 17–24.

Devanagari. (n.d). In *Wikipedia*. Retrieved July 2010 from http://en.wikipedia.org/wiki/Devanagari

Dix, A., Finley, J., Abowd, G., & Beale, R. (2004). *Human–computer interaction* (3rd ed.). London, UK: Prentice-Hall.

Dvorak, A., Merrick, N., Dealey, W., & Ford, G. (1936). *Typewriting behavior*. New York, NY: American Book Company.

Fazly, A. (2002). *The use of syntax in word completion utilities*. (Unpublished master's thesis). Department of Computer Science, University of Toronto, Toronto, Canada.

Fischer, A. R. H., Price, K. J., & Sears, A. (2006). Speech-based text entry for mobile handheld devices: An analysis of efficacy and error correction techniques for server-based solutions. *International Journal of Human–Computer Interaction*, *19*, 279–304.

Fitaly. (2009). *The one-finger keyboard*. retrieved from http://www.fitaly.com/fitaly/fitaly.htm

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, *47*, 381–391.

gate2home. (2010). *Online onscreen virtual Hindi keyboard emulator on the Internet*. Retrieved from http://www.gate2home.com/?language=hi

GIST, C. D. (2008). *Research areas—Search engine technologies*. Retrieved from http://pune.cdac.in/html/gist/research-areas/set.aspx

Google. (2009). [Indic onscreen Keyboard iGoogle gadgets]. Retrieved from http://www.google.com/ig

Grudin, J. T. (1983). Error patterns in novice and skilled transcription typing. In W. E. Cooper (Ed.), Cognitive Aspects of Skilled Typewriting (pp. 121–143). Springer-Verlag, New York.

Guruji. (2009). http://www.guruji.com/hi/index.html

Hong, T. P., Wang, H. S., Lin, W. Y., & Lee, W. Y. (2001). Evolution of appropriate crossover and mutation operators in a genetic process. *Applied Intelligence*, *16*, 7–17.

Hooke's Law. (n.d). In *Wikipedia*. Retrieved July 2010 from http://en.wikipedia.org/wiki/Hooke'slaw

IBM. (n.d.). *SPSS statistics*. Retrieved from http://www-01.ibm.com/software/analytics/spss/products/statistics/

Jelinek, F. (1990). Self-organized language modeling for speech recognition. In A. Waibel & K. F.Lee (Eds.), Readings in Speech Recognition (pp. 450–506), Morgan-Kaufmann, San Mateo, CA.

Joshi, A., Ganu, A., Chand, A., Parmar, V., & Mathur, G. (2004). Keylekh: a keyboard for text entry in Indic scripts. In *CHI extended abstracts on human factors in computing systems* (pp. 928–942). New York, NY: ACM.

Kieras, D. (1993). *Using the keystroke-level model to estimate execution times*. Retrieved from ftp://www.eecs.umich.edu/people/kieras/GOMS/KLM.pdf

Kieras, D. E. (2002). The Human-Computer Interaction Handbook. In J. A. Jacko & A. Sears (Eds.), (2nd ed., chap. Model-based Evaluation). NJ, USA: Lawrence Erlbaum Associates.

Languages of India. (n.d). In *Wikipedia*. Retrieved July 2010 from http://en.wikipedia.org/wiki/Languages_of_India

Lewis, J. R., Kennedy, P. J., & LaLomia, M. J. (1999). Development of a Digram-Based Typing Key Layout for Single-Finger Stylus Input. In Proceedings of 43rd annual meeting of the human factors and ergonomics society (pp. 419–419). Houston, Texas, USA: Human Factors and Ergonomics Society.

Lipik. (2007). A predictive text input system. Retrieved from http://www.lipik.in

LooKeys. (2009). Indian languages virtual keyboard. Retrieved from http://www.keyboard4all.com

MacKenzie, I. S., & Tanaka-Ishii, K. (2007). *Text entry systems: Mobility, accessibility, universality*. Burlington, MA: Morgan Kaufmann.

MacKenzie, I. S., & Zhang, S. X. (1999a). The Design and Evaluation of a High Performance Soft Keyboard. In Proceedings of the conference on human factors in computing systems (chi) (pp. 25–31). Pittsburgh, USA: ACM.

MacKenzie, I. S., & Zhang, S. X. (1999b). Text entry using soft keyboards. *Behaviour and Information Technology*, *18*, 235–244.

Mankoff, J., & Abowd, G. D. (1998). Cirrin: A word-level unistroke keyboard for pen input. In *Proceedings of the 11th annual ACM Symposium on User Interface Software and Technology* (pp. 213–214). San Francisco, CA: ACM.

Manning, C. D., & Schutze, H. (2002). Foundations of statistical natural language processing. *Natural Language Engineering*, *8*, 91–92.

Microsoft. (2009). Onscreen keyboard. Retrieved from http://windows.microsoft.com/en/windows7/Type-without-using-the-keyboard-On-Screen-Keyboard

Mistry, P., & Nayak, N. (2007). AKSHAR: A Mechanism for Inputting Indic Scripts on Digital Devices. In Proceedings of usid (pp. 1–11). Hyderabad, India: USID Foundation.

Mukherji, P., & Rege, P. P. (2009). Shape feature and fuzzy logic-based offline Devnagari handwritten optical character recognition. *Journal of Pattern Recognition Research*, *25*(4), 52–68.

MyLanguage. (2008). Hindi virtual keyboard. Retrieved from http://www.mylanguages.org/hindi_keyboard.php

Newslog, I. S. (2010). SMS. Retrieved from http://www.itu.int/ITUD/ict/newslog/CategoryView,category,SMS.aspx

Obitko, M. (1998). *Selection-Introduction to genetic algorithms*. Retrieved from http://www.obitko.com/tutorials/genetic-algorithms/selection.php

OmicronLab. (2010). Avro Bengali keyboard. Retrieved from http://www.omicronlab.com/avro-keyboard.html

Perrinet, J., Pañeda, X. G., Cabrero, S., Melendi, D., García, R., & García, V. G. (2011). Evaluation of virtual keyboards for interactive digital television applications. *International Journal of Human–Computer Interaction*, *27*, 703–728.

Releases Radicati Group. (2009). *Email statistics report, 2009–2013*. Retrieved from http://www.radicati.com/?p=3237

Reports, E. M. (2009). *Email and webmail statistics*. Retrieved from http://www.email-marketing-reports.com/iland/2009/08

Sarcar, S., Ghosh, S., Saha, P. K., & Samanta, D. (2010). Virtual Keyboard Design: State of the Arts and Research Issues. In Students' technology symposium (techsym) (pp. 289–299). Kharagpur, India: IEEE.

Sears, A., & Zha, Y. (2003). Data entry for mobile devices using soft keyboards: Understanding the effects of keyboard size and user tasks. *International Journal of Human–Computer Interaction*, *16*, 163–184.

Shieh, K. K., & Lin, C. C. (1999). A quantitative model for designing keyboard layout. *Perceptual and Motor Skills*, *88*, 113–125.

Shneiderman, B., & Plaisant, C. (2004). *Designing the user interface: Strategies for effective human–computer interaction* (4th ed.). Boston, MA: Addison Wesley.

Sholes, G. L., Glidden, C., & Soule, S. W. (1978). Improvement in Type-writing Machines (Nos. 79,868).

SNLTR. (2009). *Society for natural language technology research*. Retrieved from http://www.nltr.org/

Soukoreff, R. W., & MacKenzie, I. S. (1995). Theoretical upper and lower bounds on typing speed using a stylus and soft keyboard. *Behaviour and Information Technology*, *14*, 370–379.

Soukoreff, R. W., & MacKenzie, I. S. (2002). A character-level error analysis technique for evaluating text entry methods. *Proceedings of the Second Nordic Conference on Human–Computer Interaction*, 241–244.

Soukoreff, R. W., & MacKenzie, I. S. (2003). Metrics for text entry research: An evaluation of MSD and KSPC, and a new unified error metric. *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 113–120.

Statistics, I. W., & (2010). *Top ten Internet languages*. Retrieved from http://www.internetworldstats.com/stats7.htm

Technology Development for Indian Languages. (1986). *Keyboard standard*. Retrieved from http://www.tdil.mit.gov.in/pdf/KeyboardStandard2.pdf

Technology Development for Indian Languages. (2009). *iLeap—Multilingual word processor*. Retrieved from http://www.cdac.in/html/gist/products/ileap.asp

Google launches virtual keyboard. (2010). The Telegraph. Retrieved from http://www.telegraph.co.uk/

Telugu script. (n.d). In *Wikipedia*. Retrieved July 2010 from http://en.wikipedia.org/wiki/Telugu_script

Unicode Consortium. (2008a). *Unicode Bengali code chart*. Retrieved from http://www.unicode.org/charts/PDF/U0980.pdf

Unicode Consortium. (2008b). *Unicode Hindi code chart*. Retrieved from http:/ /www.unicode.org/charts/PDF/U0900.pdf

Unicode Consortium. (2008c). *Unicode Telugu code chart*. Retrieved from http://www.unicode.org/charts/PDF/U0C00.pdf

Vistawide. (2009). *Top 30 languages spoken in the world by number of speakers*. Retrieved from http://www.vistawide.com/languages/top_30_languages.htm

Wobbrock, J. O., & Myers, B. A. (2006). Analyzing the input stream for character-level errors in unconstrained text entry evaluations. *ACM Transaction of Computer Human Interaction*, *13*, 458–489.

Wright, A. H. (1991). Genetic algorithms for real parameter optimization. In G. J. E. Rawlins (Ed.), *Foundations of genetic algorithms* (pp. 205–218). Burlington, MA: Morgan Kaufmann.

Zhai, S., Hunter, M., & Smith, B. A. (2000). The Metropolis keyboard—an exploration of quantitative techniques for virtual keyboard design. *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*, 119–128.

Zhai, S., Hunter, M., & Smith, B. A. (2002). Performance optimization of virtual keyboards. *Human–Computer Interaction*, *17*, 229–269.

ABOUT THE AUTHORS

Debasis Samanta received his Ph.D. in Computer Science & Engineering from Indian Institute of Technology Kharagpur, India. He holds M.Tech and B.Tech degrees all in Computer Science & Engineering from Jadavpur University, Kolkata, India, and Calcutta University, India, respectively. Presently he is an Associate Professor in the School of Information Technology, Indian Institute of Technology Kharagpur, India. His current research includes human–computer interaction, biometric-based system security, and software testing.

Sayan Sarcar holds a B. Tech degree in Computer Science & Engineering and M.Tech degree in Multimedia & Software Systems from West Bengal University of Technology. Presently he is a Ph.D. student in the School of Information Technology, Indian Institute of Technology Kharagpur, India. His research interests include design of virtual keyboards in Indic languages multimodal text entry mechanisms.

Soumalya Ghosh received a B. Tech degree in Information Technology from West Bengal University of Technology. Presently he is an M.S. student at the School of Information Technology, Indian Institute of Technology Kharagpur, India. His research interests includes design of virtual keyboards in Indian languages.