Eyeboard++ : An Enhanced Eye Gaze-based Text Entry System in Hindi

Sayan Sarcar
School of Information Technology
Indian Institute of Technology Kharagpur
mailtosayan@gmail.com

Prateek Panwar
Computer Science & Engineering
SDMCET Karnataka
prateekpanwar31@gmail.com

ABSTRACT

Of late, eye gaze has become an important modality of text entry in large and small display digital devices. Despite many tools being developed, issues like minimizing dwell time and visual search time, enhancing accuracy of composed text, eye-controlled mouse movement stability etc. are yet to be addressed. Moreover, eye typing interfaces having a large number of keys suffer from many problems like selecting wrong characters, more character searching time etc. Some linguistic issues often decline in minimizing dwell time incurred for character by character based eye typing task. The aforementioned issues are prominently evolved in case of Indian languages for its many language related issues. In this paper, we propose a gaze-based text entry system *EyeBoard++* for Hindi, national language of India which minimizes dwell time by introducing word completion and word prediction methodologies side by side mitigates visual search time by highlighting next probable characters. Performance evaluation shows that proposed interface achieves text entry rate on an average 9.63 words per minute. As designed, the proposed interface can effortlessly be suited in medium-sized display devices like Tablet PC, PDA etc. The proposed interface design approach, in fact, provides a solution to deal with complexity in Indian languages and can be extended to many other languages in the world. Also, the developed system can be used by the people with motor disabilities.

INTRODUCTION

In recent times, eye gaze-based text entry has been evolved as an alternate interaction mechanism which supports faster text entry with less cognitive load in digital devices [20]. This method is nearly similar to other primitive text entry mechanisms, only the difference lies in interacting with human organ namely eye gaze instead of traditional hand or finger. The benefit of eye gaze-based text entry also can be interpreted as it can be extended, with same setup for able-bodied, toward disabled people who are capable of interacting visually and having good vision. Many applications are developed supporting eye gaze-based text entry ([19, 22, 35]), even for mobile devices [6].

India HCI and APCHI 2013, September 24 - 27 2013, Bangalore, India

http://dx.doi.org/10.1145/2525194.2525304

Eye gaze-based text entry, often called as *Eye typing*, in virtual keyboard is performed through direct eye pointing (i.e. looking) on the exact button in the keyboard [21]. *Eye press*, that is, key selection in this context is accomplished by fixing the gaze on the key for a slightly prolonged duration which is stated as *dwell time*. Alternatively, *eye blink* can be a way to perform gaze typing. Gaze-based text entry mechanism posses a number of design issues beside being popular among alternate text entry mechanisms [18], which make it a unique technique with its own set of research agenda.

Screen space can be saved by decreasing the number of keys and space between keys of the keyboard [22] to accommodate other applications on the screen. On the other hand, users are getting comfort in typing if the key size gets bigger applicable in a low spatial resolution setup [9]. This situation results in occupying larger screen space instead of having fewer keys present in the keyboard. So, an optimal size of the keys and space between keys need to be decided for obtaining balance between eye movement, screen space and user friendliness. In the *eye typing* process, most prominent subtask user performed is visually searching the desired character in the interface which is significantly affected by the features of the interface such as color, orientation, shape, size, spatial frequency etc. [38]. Further, screen area becomes an important constraint required to be considered specially for small display devices. Keeping this issue in mind, Špakov and Miniotas [32] designed an easily usable keyboard which saves screen space and requires no special learning.

An objective in developing dwell-based eye typing interface is to mitigate the trade-off between speed and accuracy at different levels of cognitive complexity. A long dwell time leads to the false selection of characters while shorter dwell time enhances the chance of *Midas Touch* problem [11]. So always it is not obvious that better text entry indicates less dwell time incurred. The dwell time also hinders to achieve limited typing speed (not beyond the limit of maximum) as the user needs to fixate for the dwell time before each selection. Majaranta and Räihä [21] reported that most gaze typing evaluations were conducted with novices using a constant and fairly long dwell time ($450 - 1000$ ms). Recently, Wobbrock et al. [37] used a short dwell time of 330 ms and achieved text entry rate of 7 wpm. Špakov and Miniotas [33], and Majaranta and Räihä [21] studied automatic adjustment of dwell

time. Though, human beings are not equipped with controlling eye gaze for a longer session.

Few eye typing interfaces in English have augmented character and word prediction methodologies to mitigate overall dwell time overhead faced in character by character text entry which inherently enhance the text entry rate. An eye typing interface, namely *GazeTalk* was proposed by Hansen et al. [9] which provides both letter and word prediction. The interface contains both word list and character panel containing six probable characters depending on the lastly composed character. After every character concatenation to the text being composed, the word list and character panel are updated accordingly. Mackenzie and Zhang [16] compared word and letter prediction in a gaze typing system with an implementation of both the predictions in an on-screen keyboard. They compared both the predictions and observed that letter prediction was as good as word prediction and sometimes even better.

Researchers have also started to analyze the suitability of different *gaze typing* methodologies on text entry interfaces developed for Asian languages. A few works on Chinese language were focused to develop effective interfaces accommodating large set of characters in *pinyin* [14]. To the best of our knowledge, no such work has been proposed toward developing eye typing interface for Indian language.

ISSUES IN DESIGNING EYE TYPING INTERFACE IN INDIAN LANGUAGES

East or South-east Asian languages contain complex scripts which make the character by character text entry a difficult task. The characteristics which makes the text entry task difficult are mentioned in the following. These are also applicable for gaze based text typing in Indian languages. Further, some issues related to eye movement and fixation behavior are also pointed out as follows.

- Large character set: Indian languages contain a large number of basic vowel and consonant characters (much more than Latin alphabet) and their combinations. As an example, to type in Devanagari one needs to input at least 34 consonants, 11 independent vowels, 10 dependent vowels, 8 diacritic marks, and their combinations to represent approximately 660 frequently used glyphs. As all characters could not be accommodated into small space, one solution could be to map more than one character in a single key with use of special keys, namely *Shift*, *Ctrl*, and *Alt* key, for selection of suitable characters [15]. This scenario, as a result, increases the number of eye presses required which further enhances the cognitive load.

- Normalization: Indian language data require normalization [3, 5], as there exist characters having equivalent Unicode representation. For example, the word रिज़र्व can be composed as र+ि+ज+.+र+्+व (containing 7 characters) as well as र+ि+ज़+र+्+व (consist of 6 characters) [3]. This is happened for multiple representation of characters with *nukta*. In addition to this, the use of "Zero-Width Joiner"[1] (Unicode value U200D) and "Zero-Width Non Joiner"[2] (Unicode value U200C) represent a conjunct in different ways. For example, "क्ष" (unicode sequence $U0915 + U094D + U0937$) in different forms can be represented as "क्‍ष" (unicode sequence $U0915 + U094D + U200D + U0937$) and "क्ष" (unicode sequence $U0915 + U094D + U200C + U0937$). These compositions, in fact, are not valid to compose text although they appear to be correct [5]. If proper normalization is not carried out, then searching the word written in one form will miss the words in another form [3].

- Input sequence: The ligatures in Indian languages are not necessarily written or read in a linear sequence. In other words, the writing order and the phonological order may not match in Devanagari [10, 15]. Further, the position of a character in a word may not be fixed. For example, to compose a word निर्मित user has to select the characters in the order: न+ि+र+्+म+ि+त. Note that this type of required ordering demands enough cognitive load on users.

- Typographical variants: In Hindi language, several words have multiple correct spellings and alternate representation forms [2, 3]. The character "*anuswar*" can be used as both half-na (e.g. हिंदी and हिन्दी and half-ma (e.g. मुंबई and मुम्बई [2, 3]. According to "Centre for Development of Advanced Computing"(CDAC), in Indian language some misspelled words are more significantly in use than their grammatically correct counterpart. For example, the word जांच is incorrect but is used more often than its correctly spelled form जाँच [2].

- Difficulty in forming *ligature* through gaze: the formation of a complex character (*glyph* or *ligature*) requires at least three key presses which is more with respect to English character composition (one for single small character and an extra shift to form capital character). As a result, user spends more time and effort to compose same length characters by eye gaze in Indian language than English.

- Presence of phonetically or graphically similar characters: There exist characters or their combinations which are phonetically similar (sounds alike) [29, 34] as (श, ष and स, (ऋ and री and (ई and यी and (ए and ये etc. Some of the characters are so much similar to the others in shape that there remains a finite chance of confusion, for example, between भ and म, घ and ध, ख and रव [7] etc. The above mentioned issues make the task of text entry more erroneous.

- More eye movement: Developing a virtual keyboard in Indian languages which supports single character allocation on a single button, the size of the keyboard becomes larger which yields more eye movements while composing texts. On the other hand, the required size of the keys and gap between keys hinder in accommodating all characters on a single screen at a time, particularly for medium and small

[1]Zero-width joiner, **http://en.wikipedia.org/wiki/Zero-width_joiner**

[2]Zero-width non-joiner, **http://en.wikipedia.org/wiki/Zero-width_non-joiner**

sized display devices. Moreover, for dwell-based eye typing, more character typing increases the effective time to complete the task.

- More key searching time: Visual search time to find a key in the keyboard interface in Indian language becomes higher due to increased number of characters present in the same [25]. Also, it has been observed that the eye movement incurred by users in a virtual keyboard while searching for key is very much abrupt and eyes are sensitive to sudden color contrast change among visual contents in the interface.

Moreover, an effective text entry system must include localization, error correction, editor support, feedback, and context of use [13]. These are more pertinently true in the context of developing Indian language based effective eye typing interface and indeed it is a challenge to deal with those issues.

In this paper, an enhanced gaze-based text entry interface for Hindi, the national and mostly spoken language of India, is proposed fulfilling the following objectives.

1. Virtual keyboard layout for text composition in Hindi
2. Augmenting word completion and prediction facilities toward achieving faster text entry
3. Predicting and highlighting next probable characters after tapping a key to mitigate visual search time

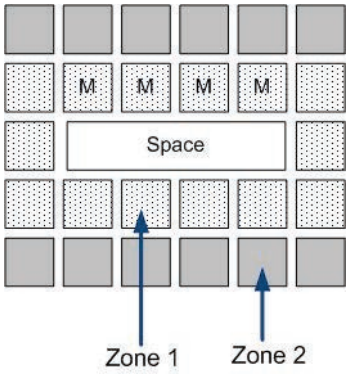
METHODOLOGY

We develop a gaze-based text entry system for Hindi language to cater the aforementioned issues. The proposed system is named as *EyeBoard++* which consists of a keyboard with Eye typing support. The proposed keyboard layout follows the design proposed by Panwar et al. [23] maintaining proper balance between three parameters namely size of the keys, space between keys and number of keys present in the layout. We propose word completion and next word prediction facilities to speed up the eye typing rate. Moreover, after tapping a character or chunk, the proposed system predicts next probable characters which lead to a valid word in the dictionary and highlights a few of them in order to decrease the visual search space of users during eye typing. In the following, we discuss our approach in details.

Developing an effective on-screen keyboard

In this work, we specifically concentrate on developing gaze-based text entry interface for Hindi. We develop eye-typing keyboard interface following the basic design principle of Panwar et al.'s English keyboard (*EyeBoard*) [23] for gaze-based text entry. These are namely, placing 27 characters (space included) in a 5×6 matrix (almost square), and holding (3^{rd} row) Space character of 4 key sizes in middle row (to increase reachability as it occurs much more than other characters in language texts) (Fig. 1(a)). The optimized values of other parameters like size of keys, space between keys and zooming of the layout are remained same in the proposed layout. *EyeBoard* layout [23] yields less eye movement than other popular English mouse or touch-based interfaces while typing as the design principle mitigates visual search of finding a key (performing visual search means moving gaze as well as mouse pointer). As a consequence, text entry speed enhances moderately. In the proposed *EyeBoard++* layout, basic design decisions about size and space between keys as well as overall zooming of the *EyeBoard* layout are followed with some changes done which are suitable for Hindi. The major modifications are stated below.

To accommodate almost 55 characters into the Hindi layout, those are distributed into two parts namely High and Low frequent character sets. We process *Wikipedia* corpus for calculating characters' frequencies. We sort the characters according to their frequencies. The first one contains the set of characters which are more frequent in the Hindi corpus whereas second part is having less frequent characters. The visualization of those parts is executed one at a time where each portion is having navigational control to move to the counterpart. At a time, only a single part can be displayed. In both the layout, position of Spacebar remains unchanged as it is most frequently occurred and thus placed at the middle portion where user's eye always move around [23]. On the basis



(a) Multi-zonal layout for virtual keyboard

(b) Developed keyboard layout (a part)

Figure 1. Schematic and implemented diagram of the *EyeBoard++* keyboard

of the fact, highly associated characters are placed surrounding to Spacebar into concentric 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 so on [23]. For this layout, the lower frequent characters are placed in second part. Characters, for both the layout part, are placed in the zones in a row-major order (means highest frequent character is placed in the leftmost top corner, the next frequent character is just right hand side of that and so on). In order to place the zone characters surrounding to previous zone (Fig. 1(a)), we also consider the unigram and bi-gram frequencies of the intra-zonal characters. In this design, we first place the characters in row-major order according to their frequencies in descending order. Then, depending on the bi-gram frequencies between each character pair, we rearrange the characters following Trial and error method. Unlike English keyboard [23], dependent vowels (*Matra*) are present in Indian languages where some of them are very frequent in occurrence. In contrast, the occurrences fluctuates much more over the corpus than vowel or consonant characters. So, it is decided to fix four keys in the zone (Fig. 1(a)) which take the *Matra*s but the decision of the eligible candidates is based on bi-gram frequencies with the last typed character at runtime. The resultant highest to lowest frequent *Matra*s are placed on the buttons following the matrix row-major order (fixed buttons are marked as M in Fig. 1(a)). As the proposed interface is developed for Indian languages, two new panels are introduced which will be opened after pressing command button on the layout (मात्राएँ for showing all *Matra* symbols and युक्ताक्षर for many commonly used *Ligatures*/complex characters). Due to space constrains the *Clear* and *BackSpace* control buttons are categorized under *Other option* button in the proposed interface (Fig. 1(b)).

Research works on touch or pointing supported virtual keyboard reveal that frequency-based key arrangement reduces visual search time as well as enhances text entry rate ([12, 26]). This design decision is followed in developing *EyeBoaed++* interface. In addition, two critical constraints regarding eye movement and fixation behavior have been analyzed to develop keyboard layouts suitable for *Eye Typing* interface; a) eyes are always tend to move, not to fix around a point for moderately large amount of time and b) sudden change in color contrast among visual contents in an interface always draws attention [24].

Augmenting word completion and prediction

To compensate dwell based low eye typing speed, we implement current word completion as well as next word prediction methodologies in Hindi. The detailed procedure is described below.

Language model

We consider Hindi *Wikipedia* corpus (written in Devanagari Unicode) to develop the resource (extracted from files available at the link `http://dumps.wikimedia.org/hiwiki/`). These pages contain texts, images and HTML tags. Since only text part was required, we applied filter to remove all non-relevant elements (any symbols with 'Virama'). The extracted texts were then "Normalized" [28] and the occurrence of "Zero-Width Joiner" and "Zero-Width Non Joiner" [28] were removed. The developed language model consists of 65000 unigram (i.e. 65000 vocabulary words and one symbol for out-of-vocabulary word, the size of file is 951KB), 1.1 million bigrams (63,284KB) and 2.2 million trigrams backoff probability (1,49,452KB). It also contains its probability mass distribution (used in a situation when one of the higher order n-gram sequence does not exist). The CMU-SLM toolkit [30] has been used to calculate backoff probabilities for Hindi *Wikipedia* resource. This language modeling tool ignores space character occurrences in the corpus. One of the general problems when using word-level language models is the dictionary problem, also known as out of vocabulary (OOV) problem. It is hard to take decisions when the composed word is not present within context. Sometimes, people consider these words as *Noun* but this is not always become a correct step. As a solution, character-based language model (upto trigram) has been developed which further helps in constructing character prediction algorithm. This type of prediction performs better than word predictors.

The *EyeBoard++* keyboard gets modified to avail the word prediction and completion facilities. As users can not use these two facilities simultaneously, a single panel is added in the keyboard layout which dynamically shows next character or word depending on within a word composition and after pressing Spacebar, respectively. The panel has the capacity of keeping six character buttons (similar with the main keyboard panel where one row contains six buttons). The fact of tending user gaze movement horizontally rather than vertically influences the user comfort in moving user eyes within the prediction list accordingly. As users' eyes are moving back and forth between keyboard and text entry area during eye typing, the prediction panel is placed maintaining similar distance in between these two components of the gaze-based text entry interface.

Completion of current word

Word completion scenario can be suitably described with an illustration. Suppose the user is typing a sentence and the sequence $\ldots w_{i-2}w_{i-1}w_{i_{prefix}}$ has been entered so far where w_{i-2} and w_{i-l} are the last two composed words, and $w_{i_{prefix}}$ is current word in composition. Let W be the set of all words in the dictionary that begin with the prefix $w_{i_{prefix}}$. A word completion algorithm attempts to select the n most-appropriate words from W that are likely to be the user's intended word, where n is, in our case, 6. The general approach is to rank candidate words $w_i \in W$ according to their probability of occurrences with the current context. In the proposed approach, word level trigram probabilities have been used for ranking (for the first and second word in the sentence, currently system takes the unigram and bigram probability sequence, respectively). The completion algorithm always checks the current character chunk and matches with words occurred after previous two words. The developed system modifies the basic working methodology to make it more robust. To avoid phonetic ambiguity, both typed and equivalent part of the target string are processed separately. There, phonetic equivalent characters, if exist in individual string, are mapped to their equivalent base character. On the other

hand, special characters (halant and other characters) are removed from the chunks. After that, phonetic scores of two chunks are calculated accordingly. In any case, if the system finds a chunk which is not matched with any words in trigram word sequence from the dictionary, it searches the bigram sequence. If it finds certain matches, the probability values are calculated with backoff and bigram probabilities. Failure in this sequence drives the system toward calculating unigram probability with certain backoff weight. Depending on the final calculated probability of each sequences, the intended word rank is decided. The scenario can be illustrated with a suitable example. Suppose, user types क्या आप अ with his eye. Existing word completion systems supporting Indian language script suggest the words whose parts have completely matched with the typed character chunk. Suppose user has typed क्या आप अंग्रेज़ी में बा with eye. The existing completion systems, even working on the word trigram, could not find any next probable characters. Instead, the proposed *EyeBoard++* interface analyzes word bigram and ranks the next probable characters starting with बा. The resultant words are बांग्ला, बांटा, बाधा, बाँटा, बाहर and बाँट (Fig. 2).

Figure 2. EyeBoard++: completion of current word

Prediction of next word

The next word prediction module activates while user presses Spacebar after manually composing a word or selecting a word from the prediction list. The word prediction methodology is having similarity with completion module as it starts working from word trigram level and if result remains unmatched after execution, bigram and unigram language modeling are also being exercised with associated Backoff weights. Also, while processing probable and typed characters, the system calculates phonetic scores of them using same procedure described in word completion module. The working scenario can further be visualized with an example (Fig. 3). Suppose, user types the word chunk as आप की यात्रा सुखद and wants to continue the sentence composition. while he presses Spacebar, next probable six words are being displayed. Among them, two (थी and और) are calculated from word trigram and others (अनुभूति, अनुभव, अंत and नहीं) are picked from bigram list taking last composed word into consideration.

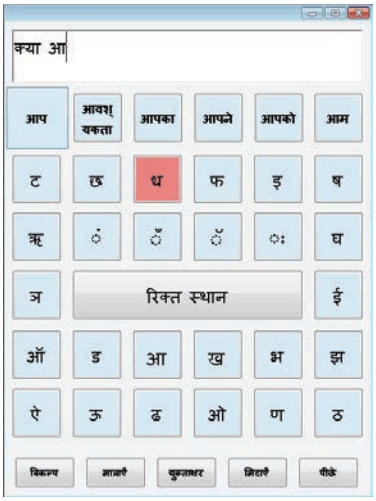
Figure 3. EyeBoard++: prediction of next word

Highlighting next probable characters

User feedback over a long period of time on different keyboard designs reveals that, for novice as well as experts, visual search task takes significant amount of time for character based text composition. Unlike to mouse or touch based interface, eye movement time includes visual search time for a gaze-based interface. From the analysis of Sears et al. [27], it has been clearly summarized that visual search time does not vary only with number of keys present on keyboard interface, rather it depends on certain other features like size of the keys, distance between keys, different color of the key groups etc [25]. Few papers in the domain of mouse or touch-based interfaces point out the importance of visual search time and try to mitigate it in mobile devices [8, 17]. In eye typing context, the eyes can be easily distracted and focused onto some different colorful keys in the interface. Using this natural phenomena, if some characters within the interface get highlighted, then user can concentrate on those rather searching the others. In this way further, the visual search time can be minimized. Applying different colors on some characters had been exercised by Mackanzie and Zhang [16] in their eye typing interface and they got improved result in terms of eye typing rate. In contrast, it has been observed from expert-based study on proposed interface that many colors actually distracts the user concentration and slows down the typing rate. Also, study conducting on different number of colors (a range of two to five) applied on next probable characters concludes that two colors given (half to the most probable and half to next most probable characters) results the optimized visual search time. The character level bigram and trigram probabilities are calculated from *Wikipedia* corpus. Initially, based on the last two consecutive characters typed, the system automatically predicts next characters. Six characters in the

whole interface, spread over two parts showing one at a time, have been displayed after each iteration. It may be the situation that at a instance, trigram analysis produces less number of characters. For this case, the mechanism further investigates the bigram frequencies and fills the deficit with higher ranked characters.It further minimizes the text entry error occurred due to presence of phonetically similar characters because the ambiguous pairs may not have higher probability values. As an example, after composing character chunk क्या आ, *EyeBoard++* chooses next probable characters as त, य, द, र and ब where first five characters are displayed in layer 1 and sixth character in the layer 2 (which we can get by pressing 'आगे' command button from the keyboard)(Fig. 4(a) and 4(b)). This scenario increases user comfort as well as diminishes error committing tendency of frequent keyboard users. Although the keyboard occupies larger area on the screen to be positioned, this method helps common users to confine their eye movements within a specific region and selects the intended character from highlighted keys, in most of the cases.

(a) Layer 1



(b) Layer 2

Figure 4. *EyeBoard++*: next character highlight

EXPERIMENTAL SETUP

After designing the proposed *EyeBoard++* eye-typing interface in Hindi obeying the aforementioned design approaches. user experiments need to be conducted to judge the efficacy of the proposed interface over existing systems. Experimental setup prepared for conducting user study is described below.

Apparatus

All experiments are conducted in a low cost eye tracking setup using 2.2GHz Intel Core2Duo processor with $15''$ wide screen LCD color monitor having 1440×900 resolution. The apparatus used in user experiments were: modified *Sony PlayStation Eye* webcam (sampling rate is 40 fps) where original lens is replaced by manual focus and Infrared (IR) filter removed lens, *IR Lamp* contains a matrix of 10 IR LED and open source *ITU GazeTracker* software [1] developed by IT University of Copenhagen. For better detection of eye gaze, we built a setup where camera was placed attached to a stand by hanging wires in front of eye at the center area of the screen. The distance between user's eye from camera and screen were approximately 8 cm and 60 cm, respectively. The developed *EyeBoard++* and other interfaces considered for comparison were written in $C\#$ using Visual Studio 2010. The key press events were recorded automatically and stored in a log file using a separate event hooking program. Another window hook program was developed to track gaze positions, also written in $C\#$. All experiments were performed in Windows 7 environment. Controlled light conditions and positioning of the setup were maintained.

Participants

Six participants (4 male, 2 female) were selected from the local area after their eye testing; among them, six were agreed. Participants' age were ranged from 25 to 35 years (mean = 28). All are daily computer users, access their Desktop or Laptops on an average 5 hours per day, but no prior experience with eye tracking. All participants have normal vision and expertise in composing text through digital devices. 5 participants are right-eye dominant and 1 is left-eye dominant, as determined using an eye dominance test [4].

Procedure

Before every eye typing session, users synchronized their eyes with the mouse pointer by performing *Calibration* task with gaze tracker so that pointer movement were controlled by eye gaze. Inability of users in moving their eyes beyond the visibility range of screen during the session was a major issue of the experiment. Alternatively, participants first wrote the phrase using pen and paper or listen while instructor prompting it. For this purpose, we collected 55 phrases having average 100 to 120 word length of each from Hindi short stories. The main objective was to compose the phrases as fast as possible without bothering the typing errors committed.

Before the experiments, participants spent first few sessions for training where they were briefed about the nature of the experiment and completed a short demographic questionnaire. They also got familiarized with eye tracking hardware (camera and Infrared lamp positions) and the *EyeBoard++*

keyboard interface along with other designs (Fig. 5). The total time for this interaction was about 10 minutes. For all user experiments, the dwell time was fixed as 500 ms.

After practicing on paper, participants were given two practice phrases with eye typing interfaces appeared at random order. First session took about one hour, and data were not considered for analysis. After completion of training, each participant on an average, composed 8 texts for testing. Before starting of each sessions, users assured the instructors about their memorability of the practiced set. Random ordering was followed for selecting a keyboard in each session. The order was counterbalanced across the participants. Each testing session took about 8 to 15 minutes which was dependent on user's proficiency on eye typing using the setup. On average, a gap of 45 minutes was maintained between two consecutive sessions with respect to single user. On an average, a user spent time in performing 3 to 4 experiments per day. Most of the users performed eye typing with all the designs in Hindi, only a few had not completed all the experiments successfully. 6 texts were selected for experiments and among these, 1 was taken from the in-domain *Wikipedia* corpus and other 5 were taken from out-of-domain texts such as novels, stories etc. for judging the design efficacy.

Dependent measures

The dependent measures used in this experiment are words per minute (WPM) and the total error rate [31, 36].

STUDY DESIGN

A longitudinal study was performed to measure the effectiveness of the proposed *EyeBoard++* design (detailed procedure discussed earlier). This was done by considering two other alternate designs for eye typing in Hindi. These designs are stated as following.

Design 1 (Multi–zonal character key arrangement for less eye movement): In this design, we initially arrange the characters into two different concentric zones surrounding center based on their unigram and bigram frequency associations. Moreover, we arrange the zonal characters based on trial and error method to achieve less eye movement while searching characters. The layout of Design 1 is shown in Fig. 1(b).

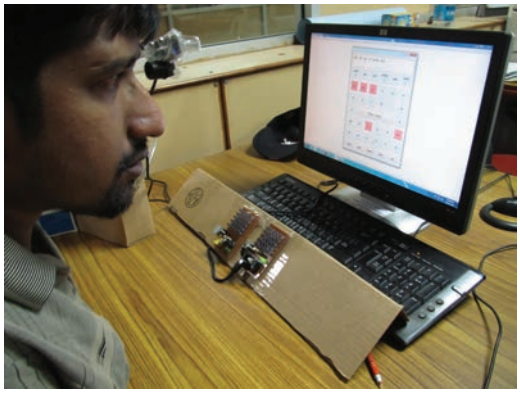


Figure 5. Participant performing experiments

Design 2 (Word completion and prediction facilities augmented with Design 1): In the basic design, character by character based text entry spends more time in dwelling on the characters and more effort it requires to correct character level errors. This design addresses the issues and provides suggestions for current words in composition and next word prediction (Fig. 3).

These two incremental designs are now compared with the proposed *EyeBoard++* design, where as an increment from *Design 2*, character highlighter module is newly added.

EXPERIMENTAL RESULTS

Several user experiments were performed to judge the effectiveness of the proposed system with respect to performance evaluation metrics namely *Text entry rate* and *Total error rate*. Two study designs along with the *EyeBoard++* design were taken into consideration for this purpose. Users were also involved for subjective evaluations with respect to *user friendliness*, *usability* etc. Data for each participant are averaged for each session to form single measures per participant per session on the basis of two aforementioned metrics [31]. Participants complete a total of 3 trials $\times$ 3 designs $\times$ 6 sessions $=$ 54 trials. With 6 participants, the entire study comprised of 324 trials.

Text Entry Rate

Summary of user experiment results (calculated by averaging all user results with 3 designs) is depicted in Fig. 6. It reveals that *Design 2* yields 65.04% better text entry rate (8.92 wpm $SD = 1.57$) than *Design 1*. Similarly, *EyeBoard++* gives, on an average, 9.63 wpm ($SD = 1.23$) which is 77.90% more than *Design 1*.

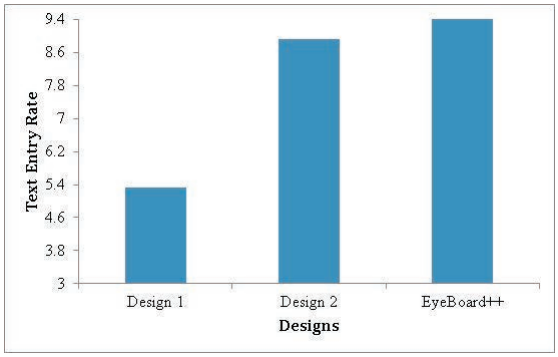


Figure 6. Comparison among different study designs

Total Errors

Over 6 sessions, the total error rate, on an average, (Fig. 7) is 12.14% for *EyeBoard++* and 13.28% and 12.72% for *Design 1* and *Design 2*-based designs, respectively. However, total error rates drop significantly over sessions ($F(5, 47) = 4.72$, $p < 0.05$).

The results we got from the above error analysis do not strictly reflect better performance of the proposed *EyeBoard++* system than other designs. The result reveals only that using the proposed interface, users left less errors uncorrected than other designs.

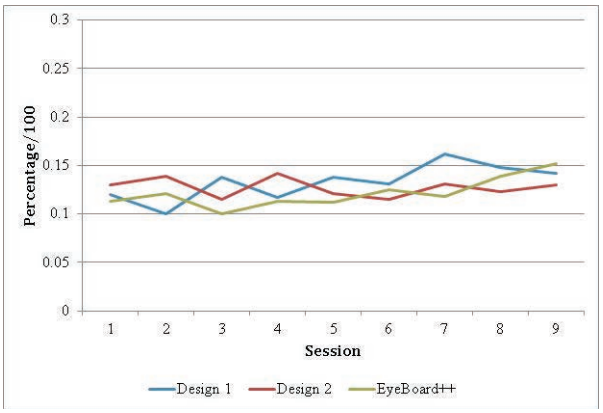


Figure 7. Comparison between total errors of 3 designs

Significantly, as there exists no such work on developing gaze-based text entry interfaces in Indian languages, we were unable to prove the design efficiency of the proposed interface by comparing it with others through user experiments with respect to same performance evaluation metrics.

Subjective Evaluation

We collect the subjective ratings from the participants with the *nonparametric Wilcoxon Matched Pairs Signed Ranks Test*. We talked with the participants before and after each session asking them about their eye strain and tiredness in 1 to 7 Likert-scale. The result reveals that users like the proposed word prediction and completion interface than other keyboard designs for ease of use ($z = 57.00, p < .001$) and less stressfulness ($z = -55.00, p < .001$). They find *EyeBoard++* interface more faster ($z = 51.00, p < .01$) and thus fun to use. However, users agreed that concentration was needed for eye typing through proposed keyboard, but they could improve their eye typing skill with practice easily. They also felt that gaze typing was clearly slower than using a conventional hardware keyboard.

The level of tiredness is quantified by subtracting the first value from the later. Analyzing the experimental results, we observe no significant difference between the average level of the tiredness, which is 0.35 in the first and 0.47 in the last session. Instead, the level of tiredness with respect to other interface accessing is lower. We also calculate the text entry speed, ease of use, and general fatigue after each session using a questionnaire with a scale from 1 to 5. An increment of text entry rate is observed (3.4 to 4.6). On the other hand, the observed ease of use, with average rating of 4.5, and general fatigue (≈ 3.5) remain approximately on the same level. Finally, participants were again interviewed after completion of the series of sessions. Participants fell that typing by gaze is fairly easy, easier than their expectations, but clearly slower than using a conventional, hand operated hardware/virtual keyboard. Moreover, operating word completion and next word prediction augmented eye typing interface was fairly easy, but after getting familiarized after spending moderate number of trials.

Participants thought that they got improvement in gaze typing over the sessions, especially in the beginning. All participants believed that incorporating dynamic nature of the interface through changing character or word level features distracted their concentration early which got minimized after performing certain number of sessions. That is why, initially, people were not using prediction rather typing character by character. Participants also admitted that the additional cost of perceptual and cognitive load, caused by shifting the focus from the keyboard to word list and repeated scanning of that list, is incurred after every eye typing session.

Threats to validity

Relating to our experimental setup, experimental procedure and experimental results, we would like to point out their validity and limitations.

- Currently, we have developed a low-cost eye tracking setup which can be easily replicated. However, the accuracy of this still is not up to the mark and thus, the applicability confines within performing experiments in controlled environments. We further have tried to improvise the situation by fixing the infrared (IR) filters within visible range and placing the camera as close to eye for more accurately detecting eye gaze during calibration phase.

- We performed user experiment with 3 keyboards and 6 participants where each participant, in training session, practiced with 2 texts per keyboard. In case of avoiding redundant text occurrence, we need to collect at least $3 \times 6 \times 2 = 36$ texts whereas, practically, we gathered 9 texts each having 6 phrases (a text consists of many phrases; total $10 \times 4 = 40$ phrases). So, while user is interested to type more than 2 texts for testing or number of participants gets increased, in current scenario, we are bound to encourage redundancy. On the other hand, sample text pool needs to be updated regularly which, in this work, was not handled.

- In the subjective evaluation section, the paper lacks in providing the questions those were asked to participants. This problem occurs because we usually asked a common question to all participants as "How do you feel after typing through proposed *EyeBoard++* as well as two other interfaces?". As the answers are usually large and general rather to any point, we summarize them in generic way.

- The typographical variance present in the Hindi alphabet makes users confused in selecting the spelling they want. Presently, we are not following a strict rule which always chooses word of a particular spelling. If followed, the number of alternate words becomes less which inherently increases the chance of other probable next candidates to be displayed in the prediction list.

Discussion

By analyzing the experimental results, it is evident that after training, users achieved faster text entry by gaze using *EyeBoard++* than others. The word completion and prediction modules help users in typing faster with gaze, same as observed in English [9, 16]. The computer proficient people, at least, are not now wasting time by searching the key as

EyeBoard++ provides next probable words beside characters This scenario proves to be moderately fast while user is comfortable with the interface. Surprisingly, it has been observed that people are not using the character highlighter as much as the word prediction and completion. The limited screen space acquired in *EyeBoard++* system also offers an advantage over off-screen targets in limiting saccade distance to the dimensions of *EyeBoard++*'s window.

CONCLUSION

There have been a number of gaze input applications in recent years even used in mobile environments. Due to inherent *Midas Touch* problem in gaze-based interfaces, dwell time is still the dominant command activation mechanism. In this scenario, the crucial factors affecting the speed-accuracy trade-off of gaze input are visual searching of the target and specifying dwell time conforming proper target selection. In this paper, we present a method that minimizes visual search time spent on the interface to compose texts. It also speeds up the text entry rate augmenting word completion and prediction. Overall, user evaluation, both based on text entry rate and subjective parameters, ensures that the newly proposed *EyeBoard++* gaze based text entry system, which is unique in its kind, is acceptable to the people and can be evolved as a strong alternate to existing text entry mechanisms.

The analysis on user error rate requires many user experiment data which we are lacking of. So, presently we are collecting data which can further lead us to a decision which we could not get in this work. Further, research can be carried out in many ways like controlling mouse speed, implementing spell and grammar checker activating through *dwelling* etc., which can improve text entry rate as well as accuracy of gaze-based text typing interfaces.

ACKMOWLEDGEMENT

We express our sincere thanks to IIT Kharagpur Research Scholars Mr. Manoj Kumar Sharma and Mr. Pradipta Kumar Saha for their significant contributions in conceptualizing the proposed methodology and building the low cost experimental setup, respectively.

REFERENCES

1. Agustin, J. S., Skovsgaard, H., Mollenbach, E., Barret, M., Tall, M., Hansen, D. W., and Hansen, J. P. Evaluation of a Low-Cost Open-Source Gaze Tracker. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, ACM (New York, NY, USA, 2010), 77–80.
2. CDAC. Indian Language Search Engine Technologies - Problems and Solutions, 2010. Available: `http://iplugin.cdac.in/search-engine.htm`, Accessed on September 2010.
3. CDAC. Problems with Existing Unicode Based Engines, 2010. Available: `http://pune.cdac.in/html/gist/research-areas/set.aspx`, Accessed on December 2010.
4. Collins, J. F., and Blackwell, L. K. Effects of Eye Dominance and Retinal Distance on Binocular Rivalry. *Perceptual Motor Skills 39* (1974), 747–754.
5. Consortium, U. South Asian Scripts-I, 2011. Available: `http://www.unicode.org/versions/Unicode5.0.0/ch09.pdf`, Accessed on January 2011.
6. Drewes, H., Luca, A. D., and Schmidt, A. Eye-gaze Interaction for Mobile Phones. In *Proceedings of the Mobility Conference*, ACM (2007), 364–371.
7. Ghosh, P. K., and Knuth, D. E. *An Approach to Type Design and Text Composition in Indian Scripts*. PhD thesis, Stanford University, 1983.
8. Gong, J., Haggerty, B., and Tarasewich, P. An Enhanced Multitap Text Entry Method with Predictive Next-letter Highlighting. In *Extended Abstracts on Human Factors in Computing Systems*, ACM (New York, NY, USA, 2005), 1399–1402.
9. Hansen, J. P., Hansen, D. W., and Johansen, A. S. Bringing Gaze-based Interaction back to Basics (2001). 325–328.
10. Ishida, R. An Introduction to Writing Systems & Unicode: A review of script Characteristics Affecting Computer-based Script Support and Unicode. Available: http://people.w3.org/rishida/docs/unicode-tutorial, 2010. Accessed on January 2011.
11. Jacob, R. J. K. The Use of Eye Movements in Human-computer Interaction Techniques: What You Look at is What You Get. *ACM Transactions on Information Systems 9*, 2 (1991), 152–169.
12. Joshi, A., Dalvi, G., Joshi, M., Rashinkar, P., and Sarangdhar, A. Design and Evaluation of Devanagari Virtual Keyboards for Touch Screen Mobile Phones. In *Proceedings of MobileHCI*, ACM (New York, NY, USA, 2011), 323–332.
13. Kristensson, P. O. Five Challenges for Intelligent Text Entry Methods. *AI Magazine 30*, 4 (2009), 85–94.
14. Liang, Z., Fu, Q., and Chi, Z. Eye Typing of Chinese Characters. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '12, ACM (New York, NY, USA, 2012), 237–240.
15. MacKenzie, I. S., and Tanaka-Ishii, K. *Text Entry Systems: Mobility, Accessibility, Universality*. Morgan Kaufmann Inc., MA, USA, 2007.
16. MacKenzie, I. S., and Zhang, X. Eye Typing Using Word and Letter Prediction and a Fixation Algorithm. In *Proceedings of the 2008 symposium on Eye tracking research & applications*, ACM (Savannah, GA, USA, 2008), 55–58.
17. Magnien, L., Bouraoui, J., and Vigouroux, N. Mobile Text Input with Soft Keyboards: Optimization by Means of Visual Clues. In *Mobile Human-Computer Interaction*, ACM (New York, NY, USA, 2004), 197–218.

18. Majaranta, P. *Text Entry by Eye Gaze*. PhD thesis, Department of Computer Science, 2009.

19. Majaranta, P., Aula, A., and Räihä, K. J. Effects of Feedback on Eye Typing with a Short Dwell Time. In *Proceedings of the 2004 symposium on Eye tracking research & applications*, ACM (2004), 139–146.

20. Majaranta, P., and Räihä, K. J. Twenty Years of Eye Typing: Systems and Design Issues. In *Proceedings of the Symposium on Eye Tracking Research & Applications*, ACM (2002), 15–22.

21. Majaranta, P., and Räihä, K. J. *Text Entry Systems: Mobility, accessibility, universality*. Eds. Morgan Kaufmann, San Francisco, CA, 2007, ch. Text Entry by Gaze: Utilizing Eye-tracking, 175–187.

22. Miniotas, D., Spakov, O., and Evreinov, G. Symbol Creator: An Alternative Eye-based Text Entry Technique with Low Demand for Screen Space. In *Proceedings of INTERACT* (2003), 137–143.

23. Panwar, P., Sarcar, S., and Samanta, D. EyeBoard: A Fast and Accurate Eye Gaze-based Text Entry System. In *Proceedings of IHCI*, IEEE (2012), 1–8.

24. Pomplun, M., Reingold, E. M., and Shen, J. Area Activation: A computational Model of Saccadic Selectivity in Visual Search. *Cognitive Science 27*, 2 (2003), 299–312.

25. Saha, P. K., Samanta, D., Sarcar, S., and Sharma, M. L. Analysis of Visual Search Features. *International Journal of Human Factors Modelling and Simulation 3*, 1 (2012), 66–89.

26. Samanta, D., Sarcar, S., and Ghosh, S. An Approach to Design Virtual Keyboards for Text Composition in Indian Languages. *International Journal of Human Computer Interaction 29*, 8 (2013), 516–540.

27. Sears, S., Jacko, J. A., Chu, J. Y. M., and Moro, F. The role of visual search in the design of effective soft keyboards. *Behaviour & Information Technology 20*, 3 (2001), 159–166.

28. Sharma, M. K. Word Prediction System with Virtual Keyboard for Text Entry in Hindi. M.s. thesis, School of Information Technology, Indian Institute of Technology Kharagpur, 2012.

29. Shilpa. Swathanthra Indian Language Computing Project. Available: **`http://smc.org.in/silpa/Soundex`**, Accessed on March 2012.

30. SLM, C. The CMU Statistical Language Modeling (SLM) Toolkit. Available: **`http://homepages.inf.ed.ac.uk/lzhang10/slm.html`**, Accessed on January 2010.

31. Soukoreff, R. W., and MacKenzie, I. S. Metrics for text entry research: an evaluation of msd and kspc, and a new unified error metric. In *Proceedings of the conference on Human factors in computing systems*, ACM (2003), 113–120.

32. Špakov, O., and Majaranta, P. Scrollable Keyboards for Eye Typing. In *Proceedings of the 4th Annual Conference on Communication by Gaze Interaction* (Prague, Czech Republic, 2008), 63–66.

33. Špakov, O., and Miniotas, D. On-line Adjustment of Dwell Time for Target Selection by Gaze. In *Proceedings of the third Nordic conference on Human-computer interaction*, ACM (2004), 203–206.

34. Thottingal, S. Soundex codes for Indic languages. Available: **`http://thottingal.in/soundex/soundex.html`**, Accessed on March 2012.

35. Urbina, M. H., and Huckauf, A. Dwell Time Free Eye Typing Approaches. In *Proceedings of the 3rd Conference on Communication by Gaze Interaction* (2007), 3–4.

36. Wobbrock, J. O., and Myers, B. A. Analyzing the input stream for character-level errors in unconstrained text entry evaluations. *ACM Transactions on Computer-Human Interaction (TOCHI) 13*, 4 (2006), 458–489.

37. Wobbrock, J. O., Rubinstein, J., Sawyer, M. W., and Duchowski, A. T. Longitudinal Evaluation of Discrete Consecutive Gaze Gestures for Text Entry. In *Proceedings of the Symposium on Eye Tracking Research & applications*, ACM (2008), 11–18.

38. Wolfe, J. M. Guided Search 2.0 - A Revised Model of Visual Search. *Psychonomic bulletin & review 1*, 2 (1994), 202–238.