Performance of Accessible Gesture-Based Indic Keyboard

Pabba Anu Bharath, Charudatta Jadhav, Shashank Ahire, Manjiri Joshi, Rini Ahirwar, Anirudha Joshi

Performance of Accessible Gesture-based Indic Keyboard

Pabba Anu Bharath[3], Charudatta Jadhav[2], Shashank Ahire[1], Manjiri Joshi[1], Rini Ahirwar[1], Anirudha Joshi[1]

[1] Indian Institute of Technology, Bombay, [2]Tata Consultancy Services, Mumbai, [3]PDPM-IIITDM , Jabalpur, Madhya Pradesh.
bharath.pabba@gmail.com, charudatta.jadhav@tcs.com, ahire.shashank@iitb.ac.in, manjirij@iitb.ac.in, rini.ahirwar@gmail.com, anirudha@iitb.ac.in

Abstract. Though several keyboards for Indic languages are available on Android Play store, few are accessible by the visually impaired. Particularly, none of the gesture-based keyboards are accessible. We developed an accessible prototype of the popular gesture-based, logically organised Hindi keyboard *Swarachakra*. In this paper, we present findings from a two-part study. In the first part, we conducted a qualitative study with 12 visually impaired users on *Swarachakra*. In the second part, we conducted a longitudinal, within-subject evaluation comparing *Swarachakra* and Google Indic keyboard. At the end of the two-week long study, 10 participants had spent an average of 6.5 hours typing, including training and text input tasks. Our study establishes benchmark for text input speeds for Indic languages on virtual keyboards by visually impaired users. The mean typing speed on *Swarachakra* was 14.53 cpm and that on Google Indic was 12.79 cpm. The mean speeds in last session were 21.72 cpm and 18.36 cpm respectively. Regression analysis indicates that the effect of keyboard was significant. In addition, we report the user preferences, the challenges faced and qualitative findings that are relevant to future research in Indic language text input by visually impaired users.

Keywords: Accessibility, Indic text input, visually impaired, longitudinal study

1 Introduction

Smartphones today are more than just mobile telephones. They are flexible devices that provide a wide array of information and communication services. Text input is a common activity required by many of these services. Touchscreens keyboard is the most common mode of text input on smartphones available in the market today. While this phenomenon has been of great help to most mainstream users, it is not an easy or smooth transition for those who are visually impaired [20]. Visually impaired users were reasonably comfortable with traditional keypad-based text input as it provided them tactile feedback as they typed. For them, using a virtual keyboard on a smooth, featureless screen has been a challenge.

Text input by the visually impaired using touchscreen keyboards is slow and laborious. While much work has been reported about text input on virtual keyboards by sighted people, comparatively less research has been reported about text input by visually impaired. For English, and for several other languages that use alphabetical scripts, virtual keyboards have been made accessible to an extent by "explore-by-touch" and "lift-to-type" [15] interaction. In the accessible mode, as the user moves his finger over the keyboard, the screen reader reads out each character under the finger. When the user lifts the finger, the character last read is typed, and is confirmed in a different voice.

However, a majority of Indic scripts are abugidas. Abugidas have a more complex script structure compared to the alphabets. In these scripts, a new glyph is often formed when a vowel modifier attaches itself to a consonant or when two or more consonants combine together to form a conjunct. These scripts also have about twice the number of characters than the alphabetical scripts. Several mainstream keyboard developers (such as CDAC InScript [11], Swiftkey [13] and Swype [14]) have responded to this problem by adopting the base QWERTY layout and relegating a large number of keys on one or more layers of shift. This adds to the already heavy cognitive load on users.

In recent years, "gesture-based" keyboards (such as *Swarachakra* [2] and *Sparsh* [12]) have attempted to alleviate this problem to some extent. These keyboards ease the interaction and reduce the cognitive load on the users by a combination of logical layout, dynamic pop-ups and gesture-based input. Unfortunately, this combination of gesture and pop-up is not compatible with screen readers, making these keyboards inaccessible. In prior work [1], we had presented an accessible version of *Swarachakra*. In this paper we develop this work further. We first did a qualitative study of *Swarachakra* with 12 visually impaired users. Based on the feedback, we redesigned the prototype. We then did a longitudinal empirical evaluation to compare the performances of this prototype with Google Indic [3]. Our work is the first such evaluation for Indic text input by visually challenged users. It establishes benchmark data for future research.

In the next section, we describe the challenges in the designing accessible Indic keyboards. In the third section we describe the designs of Google Indic and *Swarachakra*. In the fourth section we present the studies followed by the results. We conclude by presenting the lessons learnt and the possibilities of future work.

2. Background

2.1 Text Input by Visually Impaired

In the last few years, research on English text input on virtual keyboards by sighted users has advanced well. Using a combination of techniques such as word prediction, auto correction and shape writing [8], researchers have reported text input speeds ranging from 40 to 68 words per minute (wpm) or about 200 to 350 characters per minute (cpm) [7,8,9]. These are comparable to the speeds people achieve on desktop computers using a regular tactile keyboards.

In contrast, the progress on text input for visually impaired users has been limited. Speed in particular seems to be the problem. Azenkot et al [6] describe Perkinput, an eyes-free text input mechanism in which the user taps Braille-like patterns on a touchscreen device. They report average a peak speed of 7.36 wpm (about 37 cpm) with about 3.5 hours of practice for Perkinput, which was an improvement over the 4.52 wpm (about 23 cpm) that users could achieve using VoiceOver. Nicolau et al [26] report mean last session speed of 4 wpm (20 cpm) for Portuguese on QWERTY. Bonner et al. describe No-look notes, in which characters are arranged around the screen in an 8-segment pie menu [5]. Each segment of the menu contains multiple characters, such that all 26 letters of the English alphabet appear. The eight groups of characters (ABC, DEF, GHI, JKL, MNO, PQRS, TUV and WXYZ) correspond to the international standard mapping on a phone keypad [10]. In their study, they found the overall text entry speeds were 0.66 wpm (about 3 cpm) for VoiceOver and 1.32 wpm (about 7 cpm) for No-Look Notes. Thus, in these studies the speeds reached by visually impaired users are reported to be between 7 and 11% of typical speeds of sighted users.

By the year 2020 about 31.6 million people are estimated to be visually impaired in India [24]. To the best of our knowledge, no research on Indic text input by the visually impaired reports benchmark text input speed.

2.2 Text Input in Indian Languages

As discussed above, most Indic scripts are Abugidas, and are structurally different from alphabets. The Devanagari script [23] is used by many Indic languages including Hindi, Marathi, Konkani and Sanskrit, and is based on Brahmi [23]. Like many Indian scripts, Devanagari is structured according to the part of the body that produces the sounds.

Text input in Indic scripts has many challenges [2][4][19][25]. Often two or more keystrokes are needed to input a glyph. The most common construction is the combination of a consonant (e.g. **प**) and a dependent vowel modifier (e.g. ◌ो), which gives a single glyph (e.g. **पो**) (C+V). Conjuncts formed by joining two or more consonants (C+C) (e.g. **स + ट = स्ट**) have been particularly difficult to input. These are formed by separately inserting a joiner (halant character ◌्) between the two consonants, thus taking at least three keystrokes. Users often get confused with the sequence of characters to be typed to achieve a desired conjunct. At times the halant is clearly visible in the visual representation glyph (e.g. **ट + व = ट्व**) and in these cases, the sequence of keys is obvious. However, many common conjuncts are represented as a single glyph with varying levels of visual similarity with the original consonants (e.g. **स + त = स्त; क + र = क्र; र + क = र्क; क + ष = क्ष**).

Some other keyboards like Google Indic [3] have a different layout (as described below). These are not constrained by the basic QWERTY layout and hence have significantly reduced the need to use shift, though the other challenges remain.

Gestures have been used to improve speeds on virtual keyboards for English. Swype [14] and Swiftkey [13] are examples of one type of gestures. In these keyboards, the user moves the finger from one key to the next without lifting it, and the keyboard disambiguates the "shape" thus produced to guess the word user is trying to type. This option is also available in Indic versions of these keyboards. However, studies thus far have not reported improvements in speed in comparison with similar keyboards without

this feature (e.g. [2]). In any case, this feature is not accessible by the visually impaired for any language yet.

As mentioned above, keyboards such as *Swarachakra* [2] and *Sparsh* [12] also use gestures. Conceptually, these gestures are of different type than in Swiftkey or Swype, and are particularly relevant to text input in Indic scripts. On these keyboards, when the user touches a consonant, a pop-up appears. The popup displays glyphs that will result after adding vowel modifiers to the selected consonant (Fig. 1). The user chooses a vowel modifier by sliding his finger. With practice, the touch and slide becomes a single gesture. Thus, the user starts inputting two characters in one stroke, thereby improving speeds.

Fig. 1. Popup displaying combinations of consonant + frequent vowel modifiers in *Swarachakra* and *Sparsh* keyboards.

Benchmark data for text input in Indic scripts is less compared to English. In a recent study, Dalvi et al [25] report a between-subject longitudinal evaluation with sighted users using InScript, Swiftkey, *Sparsh* and *Swarachakra*. In their study, after about 5 hours of practice, novice sighted users reached mean speeds of 38 to 45 cpm. In our subsequent work, which involved accelerated learning and extensive practice on CDAC InScript keyboard, expert sighted users could achieve mean speeds of 110 to 120 cpm [19]. To the best of our knowledge, there are no published studies reporting benchmark speeds using the Google Indic keyboard.

3. The Keyboard Designs

3.1 Google Indic

Given its suitable design, we chose Google Indic as the representative of non-gesture based keyboards. It was also the only other Indic keyboard we found that was accessible. Google Indic keyboard is logically organised, i.e. it uses the Devanagari script structure in its layout (Fig. 2a). It has five rows of keys. It displays most of the frequently used Devanagari characters on the unshifted layer. Frequent independent vowels and corresponding vowel modifiers are in row 1 of this layer. This row shows

independent vowels by default (Fig 2a). The consonants and frequent diacritic marks are in rows 2 to 5 in a layout that closely resembles the Devanagari script structure.

When a consonant or a conjunct is typed, the first row changes to show the (C+V) form of the last-typed consonant / conjunct (Fig 2b). This makes it easier for the user to input the typical C+V glyph. While independent vowels do follow consonants, such occurrences are infrequent (e.g. the word **कई**). If the user wishes to type an independent vowel after a consonant, he needs to tap and hold the corresponding vowel key, and select the independent vowel from a pop-up. Similarly, the key for diacritic *anusvār* (ं) shows some of the frequent diacritics in the pop-up. Google Indic keyboard has two shift buttons. One shift toggles the between the characters and numerical keys. A second shift cycles through the second and third layer of the keyboard which contain some of the less frequent vowels, conjuncts and diacritics. The Google Indic keyboard is accessible by the visually impaired users by "explore-by-touch" and "lift-to-type" interaction described above.

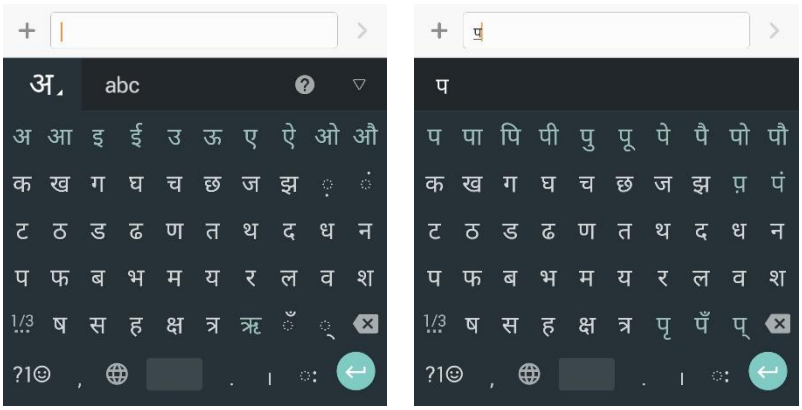

Fig. 2a Before typing a consonant, **Fig. 2b** (C+V) form of last typed consonant **प**.

3.2 *Swarachakra* Accessible

Layout of the mainstream version of the *Swarachakra* keyboard (Fig 3a) is also based on the logical structure of Devanagari script, phonetically grouped and arranged in a grid similar to those found in most school textbooks [2]. Consonants, some frequent conjuncts and vowels are arranged in a 5 x 8 grid in the middle of the keyboard. The column to the left of this grid has frequent punctuation marks, while the column to the right of the grid has backspace, diacritics and modal keys. The row below the grid has shift, space, enter keys.

The glyph on the face of the key can be entered with a tap. In the mainstream version of *Swarachakra*, the tap reveals a popup *chakra* (circular menu) with 10 options. If the key is consonant or a semi-vowel, the pop-up shows 10 frequent C+V combinations for the consonant. The user can select any of these with a slide gesture. Vowel modifiers are ordered sequentially in the *chakra*. The angle of each the vowel modifier is the same across all consonants (Fig. 3b and 3c). With practice, the vowel positions become a part

of the muscle memory of the user and he does not need to look at the *chakra* to decide the direction of the slide any more. In this case, the tap-slide gesture becomes more of a fling gesture in which two characters are input with one stroke.

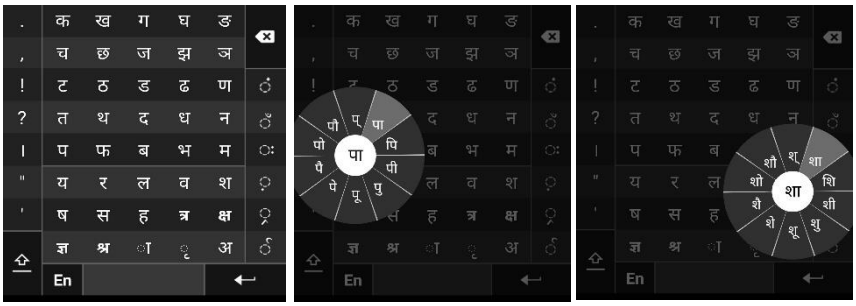

Fig. 3a. *Swarachakra* keyboard layout, **Fig. 3b.** and **Fig. 3c.** show *chakra* remains same for all consonants

The problem with the mainstream version of Swarachakra is that the pop-up prevents the users from using the "explore-by-touch" and "lift-to-type" interactions that are common to other keyboards. We developed an accessible version of *Swarachakra* [1] (Fig. 4). This version supports the "explore-by-touch" interaction, wherein the user can continuously slide his finger over the keyboard to locate the desired key as the screen reader reads out the key under the finger (Fig 4b). As the default Android Google TTS [21] does not provide feedback for some Indic characters, we used e-Speak TTS [20] for voice feedback in our prototype. If only a consonant is desired, it is input by the usual "lift-to-type" interaction, after which the screen reader reads out the input in a second voice.

As we want the users to be able to explore the keyboard by touch, this version of *Swarachakra* does not display the *chakra* pop-up on the first touch. Once the user has found the desired consonant, he can activate the *chakra* pop-up by putting a second finger down anywhere on the screen. The *chakra* pops up under the second finger (Fig. 4c). Thereafter, the user can lift the first finger. To pick one of the glyphs from the popup *chakra*, user slides the second finger in the direction of the desired vowel modifier (Fig 4d). As he does so, the screen reader reads out the vowel modifier under the finger. If the user has not found the desired vowel modifier yet, he can explore the *chakra* without lifting the second finger. *Chakra* can be dismissed without any input by tapping the first finger, while holding down the second finger

In the mainstream version of *Swarachakra, chakra* appears under the first finger, aligned to the centre of the key. However, in the accessible version, the *chakra* appears under the second finger. The user can choose where he taps the second finger. This opens up the possibility of letting the edge keys also be the ones that need a *chakra*. In the accessible version of *Swarachakra*, the left column with punctuations is dropped (Fig. 4a). This separates the keys out in the horizontal direction, and lets the users use the edges of the screen for orientation. For additional orientation, the keyboard gives strong vibrations on five keys, namely the four corners and centre of the grid (the keys for **क, ङ, ज्ञ, अ** and **द**). The users can input the spacebar and backspace through three-finger swipe gestures.

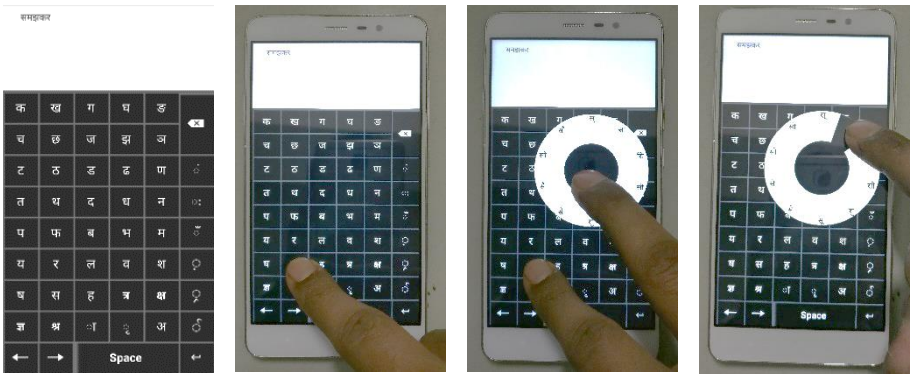

Fig. 4a accessible version of *Swarachakra*, **Fig. 4b** explore by touch to locate key, **Fig. 4c** second-finger tap to activate *chakra*, **Fig. 4d** lift the first finger and slide the second finger to make a selection

4. User Studies

4.1 Short Qualitative Study

We first conducted a short exploratory qualitative study with 12 visually impaired participants. Only two of these participants were familiar with touchscreens. Participants were explained the layout of the accessible *Swarachakra* and were familiarized with the use of keyboard in accessible mode with screen-reader. Next they were asked to type 10 words. After that, feedback was collected from the users.

Participants found the layout of the keyboard easy to learn. Most of the participants liked the combination of second-finger tap and slide gesture to select the vowel modifier. One of the most obvious issue participants faced was distinguishing between the long and short vowels in the speech output of screen reader. For example, they found it difficult to differentiate between **कि** (pronounced as kɪ as in kid) and **की** (ki: as in key), or between **कु** (kʊ as in cook) and **कू** (ku: as in cool). They also faced some difficulty in differentiating between some phonetically similar consonants (**त-थ, श-ष, ब-भ, ट-ठ, ड-ढ** etc.) in the speech output of screen reader. We also observed that keys on the screen edges were easier to locate. The backspace key in our prototype was small and participants found it difficult to use.

We redesigned the keyboard based on the above feedback. We replaced the default text-to-speech engine of screen reader with custom text-to-speech engine Lekha available via Vocaliser app [17]. We used pitch variation to differentiate between short and long vowels. We shifted the diacritics in first row to the right edge and increased the height and width of the backspace (Fig. 4a). If the user dwelled on a consonant key for longer than one second, a word starting with the consonant was read out as feedback. For example, **'तरबूज'** for **त**, '**सेब**' for '**स**' and '**धनुष**' for **ध**. This feedback was the equivalent of reading out 'Tango' for T in English keyboards.

4.4 Quantitative Longitudinal Study

We conducted a within-subject study to compare the accessible version *Swarachakra* and Google Indic keyboard. Our protocol for the study is similar to the protocol suggested by [4], except that our study had a within-subject design. The keyboard sequence was counter balanced across users.

On the first day, the user was assigned the first keyboard. The user was trained on that keyboard. During the training session, the user explored the layout of the keyboard with the help of the moderator for 5 minutes. Next, he was asked to type 8 Hindi words (Table 1). These words were representative of the typing tasks that the user might face, and were presented in the increasing order of difficulty. To ensure that the results were comparable across keyboards, the users were not trained on the "prediction" features of Google Indic keyboard. Users were trained on all other aspects of the keyboards including space, shift, backspace, touch and hold delay, etc. In talkback mode, volume up / down keys can be used to move the cursor left / right. Users were trained to use this interaction for Google Indic keyboard. However, this feature was not available while vocaliser app, and was skipped for *Swarachakra*. After the training, the user was given time to explore the keyboard on his own for 5 minutes.

After that, the user was given first-time usability test (FTU). In this, the user typed 14 Hindi words. Some examples are shown in Table 1. Again, these words are representative of the typing tasks that the user might require while typing Hindi. In the first attempt of each FTU word, no hint was provided. If the user could not type the word correctly, the user was allowed a second attempt. In the second attempt, predetermined hints were provided if the moderator deemed it necessary. After the user had typed the word, he was given feedback about how fast and accurately he had typed. If the user could type substantial number of words in the FTU, he was deemed to be trained. This ended the session on the first day.

The next day onwards, a longitudinal usability test (LTU) was started. This consisted of 10 LTU sessions, each of which required the user to transcribe eight phrases (80 phrases in alls). The phrases were a mix of Hindi idioms and phrases from school text for Hindi [16]. Example phrases are shown in Table 1. They were classified as easy, medium and hard using the classification used by Dalvi et al [4]. To provide early success to the users, we asked the users to type a random selection of eight easy-to-type phrases in the first LTU session. The difficulty of phrases was ramped up gradually till the fourth LTU session. In subsequent LTU sessions, phrases were presented randomly.

The user could touch the top left corner of the screen to hear the word / phrase to be typed and touch the right corner of the screen to hear the word / phrase typed so far. At times, the spelling of words used in the phrases to be transcribed was not clearly distinguishable in the screen reader speech output. If asked, the moderator clarified the spelling (but not in the phrase typed by the user). Other than that, no other help was provided during the LTU tasks. Like in the FTU, the user was given feedback about his speed and accuracy after completing each phrase.

Table 1. Sample of words and phrases used in the study**.**

Training words	FTU words	Sample phrases
समझकर	**कमल**	**मना कर देना**
ग़ायब	**विनय**	**ग़ायब हो जाना**
स्वागत	**ज़मीन**	**चार चाँद लगना**
पृथ्वी	**आस्तीन**	**हड़प लेना**
कार्यशाला	**बर्फी**	**अंधेरे में तीर चलाना**
अहंकारी	**नम्रता**	**इज्जत मिट्टी में मिलाना**
ऊपर	**कृपया**	**ईंट का जवाब पत्थर से देना**
चाँद	**दुःख**	**आमदनी अठन्नी और खर्चा रुपैया**

For the sake of consistency, all tasks were done using Xiaomi Redmi Note 3 device. It has a 5.5-inch screen with 1080 × 1920 pixel display and a capacitive touchscreen. To accurately log the text entered by users and the time taken, we created a software application. To ensure that users do not get fatigued, we restricted each user to a maximum of two LTU sessions per day, and provided a gap of at least 30 minutes between sessions.

During an LTU session if the moderator observed that the user was not using the keyboard well, he asked the user to explore the keyboard on his own before or after the session. The moderator provided additional training if necessary during these self-explore sessions. All these self-explore sessions were logged and were counted as "practice time". With this level of training, we were hoping to minimise errors and increase speeds.

Each keyboard attempt took between six to eight days. Once the user had completed the first keyboard, the same process was repeated for the second keyboard. Each participant was compensated a gift equivalent to ₹ 450 (USD 7) for their time.

4.5 Participants

Ten volunteers participated in the longitudinal study. Five of the users were partially blind and five were completely blind. All except one user were right-handed. Users were aged 17-28, with mean age of 22; there were 7 males and 3 females. Most of them had been using mobile phones for less than a year and the mean phone usage duration was about a year. Eight of the ten users had used Android phones. Of these, five had used TalkBack. One of these was a current iPhone user but had used Android before. One person owned a feature phone and had never used an Android phone. One person had never used a phone. None of the volunteers had any experience with typing in an Indian language.

5. Results

5.1 Practice time, Accuracy, Error Rate and Speed

During the longitudinal study, 10 users typed 80 phrases on each of the two keyboards, resulting in 1,600 phrases in total. The experiment involved about 6.5 hours of typing time for each user. The average time spent by users on a keyboard during training and self-explore sessions is an indicator of the learnability of the keyboards. Mean practice time (training + self-explore sessions) was about 1.5 hours per keyboard and mean overall typing time, including practice time, was about 3 hours (Table 2). The pauses between phrases and sessions were ignored.

Table 2. Mean practice time and overall typing time for Google Indic and *Swarachakra*

	Google Indic	*Swarachakra*
Mean training + self-explore time (mins)	89	102
Mean typing time (mins)	193	198

While speed is the primary endpoint of this experiment, speed is comparable across keyboards only when the accuracy of the typed matter is reasonably high on both keyboards. The users were trained extensively to ensure high levels of accuracy (or low uncorrected error rates). Damerau–Levenshtein distance was used as a measure of accuracy. For each session of each user, % session accuracy was calculated using the following formula:

$$\%\ Session\ Accuracy\ = \left(1 - \frac{Edit\ distance}{No.\ of characters\ to\ be\ typed}\right) * 100$$

Table 3 presents a summary of the mean and median % session accuracy for the two keyboards.

Table 3: Mean and median % session accuracy

	Google Indic	Swarachakra
Overall mean (n = 10)	94.38	95.78
Mean of last two sessions	93.88	95.10
Overall median (n = 10)	94.80	96.17
Median of last two sessions	93.92	94.71

The median session accuracy considering all 10 users for the two keyboards was quite high (95.67%). This indicates that the users were satisfactorily trained and motivated sufficiently to type accurately. High accuracy was maintained even as the users attempted to achieve high speed.

Data about "corrected error rates" is also of interest in text input research. Users had typed 25,381 characters including wrong characters and backspaces. Backspace constituted 6.32% of these and backspaced characters were another 6.39%. Fig. 5 shows the corrected and uncorrected error rates for the two keyboards as discussed in [19]. We can see that the total error rates are moderate in the first session. The corrected error rate is higher than the uncorrected, and is between 10–13% on average. It drops to between 4–7% by the 5th session, and stays in that range till the 10th session. While there is some room for improvement in the corrected error rates, these are not excessively high. We can consider that the users were reasonably trained, and hence their speeds can be compared.

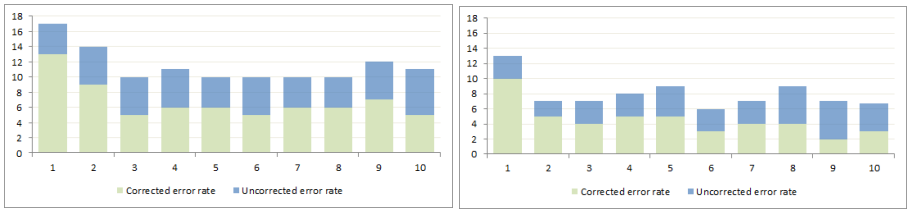

Fig. 5. Corrected and uncorrected error rates for Google Indic keyboard and *Swarachakra* Accessible keyboard respectively.

Table 4 shows mean typing speed for the first attempt, mean typing speed for the second attempt, overall mean typing speed and mean typing speed for last session, in characters-per-minute (cpm). Plot of mean session-wise cpm (Fig. 6) shows that the speed increased as users gained practice on both keyboards.

Table 4. Mean typing speed (cpm) for sessions in first attempt, second attempt, mean cpm across two attempts and mean cpm for last session for each keyboard.

Mean typing speed (cpm)	Google Indic	Swarachakra
Attempt 1(n=5)	9.8	11.2
Attempt 2(n=5)	15.8	17.9
Overall (N=10)	12.8	14.52
Last session (N=5)	18.4	21.7

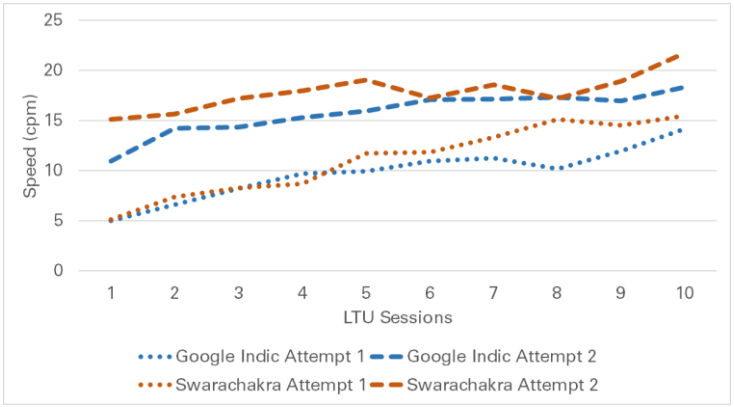

Fig. 6. Mean session speed (CPM) for all sessions of attempt 1 and attempt 2

A within-subject, repeated measures ANOVA is often used to compare the performance of the two keyboards. Although it seems that speeds were higher on *Swarachakra* for the first attempt, a repeated measures ANOVA of session-wise mean speeds showed that the differences in the mean cpm for sessions 1-10 between *Swarachakra* and Google Indic were not statistically significant, $F(1, 8) = 0.672$ ($p = 0.436$), partial $\eta^2 = 0.077$. However, we argue that in the particular case when the users are novice to text input, repeated measures ANOVA may not have sufficient power to isolate the real difference between performances of two input methods.

In a typical within-subject study that aims to compare performance of two keyboards, users are asked to perform text input tasks on the two keyboards. The order of presentation of the keyboards is counter-balanced to mitigate the effect of any transfer of learning from one keyboard to another. The unstated assumption therein is that the users are familiar with "text input" in that script in general, and are therefore already familiar with the notions and the concepts inherent to "text input". In reality, the user does gain additional "text input practice" while doing tasks on the method attempted first. This additional effect is deemed to be negligible compared to the overall text input practice that the user may have had "in life" before the experiment. Hence, repeated measures ANOVA only measures the difference in performance between the two keyboards.

However, this argument is not valid in cases such as ours where users are new to text-input as a task. Our users were unfamiliar with common text input tasks such as the use of shift, backspace, and aspects inherent to Indic text input in particular, such as vowel modifiers, conjuncts etc. In their first attempt, they were in fact learning these aspects of text input as well as the specific input method. By the time they were in their second attempt, the "general text input" practice gained during the first attempt could be quite substantial compared to their earlier "in life" practice. If that indeed is the case, the true difference of interest (the difference of speed between the two input methods) may be overshadowed by the larger difference between the two attempts. While using a repeated measures within-subject ANOVA, there is no way to separate the two differences.

To test this hypothesis, we evaluated the difference between attempts (irrespective of the input methods) and found that this was indeed true for our study. The mean session speed irrespective of input methods in attempt 1 was 10.49 cpm (N= 10, SD = 2.57) and for attempt 2, it was 16.83 cpm (N = 10, SD = 5.95). A two-sample t-test assuming unequal variance showed that the difference was statistically significant (p = 0.009).

To evaluate the relative performance of input methods while accounting for the text input practice gained in attempt 1, we performed a multiple linear regression. In our study, the mean typing speed of a given session is dependent on three factors: the attempt (first or second), the session number (1 to 10), and the input method (Google Indic or *Swarachakra*). To eliminate the effect of other variables, we performed a multiple linear regression assuming the mean session speed as the criterion variable and the attempt number, session number, and a dummy variable for the input methods (Google Indic or *Swarachakra*) as predictor variables. The model was significant and returned these values: R = 0.642, ^{R2} = 0.412, adjusted R^2 = 0.403, F = 45.707, p < 0.0005. As anticipated, all three variables emerged as significant predictors. The speed increases by 6.344 cpm per attempt (95% CI from 5.004 to 7.685, p < 0.0005) and by 0.781 cpm per session (95% CI from 0.547 to 1.014, p < 0.0005). The input method also emerged as a significant predictor (p = 0.012) with *Swarachakra* adding to the speed by 1.732 cpm (95% CI from 0.391 to 3.072), or about 11% gain over Google Indic.

The variance inflation factors (VIFs) of all predictor variables are below 2, indicating that there is no multi-collinearity among the predictor variables of the model. The distributions of the criterion variable and the z residuals were found to be normal.

5.2. Qualitative Findings

Screen-reader issue: As discussed earlier, some common issues were attributable to the limitations of the screen reader. Distinguishing between short / long vowels and similar sounding consonant pairs as read out by the screen readers was a constant cause of confusion. While we attempted to vary the pitch to differentiate between such pairs of vowels and give example words consonant, this solution was still not effective, especially at higher speech rates. This we believe is a larger issue of visually impaired users learning spelling in Indian languages, and goes beyond the scope of this text input paper.

Accidental touch with gestures: Accidental touches while using gestures could potentially lead to unintended text input and was a cause of concern. Hence, many users preferred not using gestures for space bar and backspace in *Swarachakra*. Similarly, they did not use navigation gestures in the typed text on Google Indic. While correcting errors most users chose to backspace the text all the way up to the erroneous input and then continue typing rather than using navigation gestures to the locate the error and correct it.

Findings related to keyboard designs: Most of the users were new to text-input and to touchscreen devices. The following findings must be viewed with this in mind.

Some findings are valid for design of both keyboards. Initially the users would start exploring keyboards from the left edge, moving downwards to locate the correct row. Then they would start moving along the row to locate the desired key. By the end of first couple of LTU sessions, users had learnt the keyboard layout. They tried to touchdown directly on the key, re-adjusting the finger to reach the intended key, if needed. This reduced time taken to locate keys. This strategy was used to locate space and backspace keys as well.

The keyboards being compared in our study do not give any feedback when a row boundary is crossed. While trying to explore the keys in a row, users believed they were moving perfectly horizontally, when in fact they were sometimes moving at an angle. This initially lead to some confusion about the sequence of keys in the row.

On *Swarachakra* users took some time getting used to the two finger gesture to invoke the *chakra*. Finger tap to dismiss *chakra* was found to be useful. The key with independent vowels was located at bottom-right corner of keyboard. This made second finger tap a little tricky. While training users on the *chakra* in *Swarachakra*, they were told to map the vowels onto a clock. As a result, after a few sessions users learnt the angles of the vowel modifiers.

In Google Indic keyboard, for *anusvār* (ं) key on the right edge, the popup is shown without a delay. This is inconsistent with respect to other keys, where the pop-up with additional options appears after a delay. Due to this many users avoided using the right edge to orient themselves. Sometimes users accidentally activated one of the shift keys in Google Indic and found it difficult to navigate back. In Google Indic keyboard if the users strayed onto the prediction bar at the top of the keyboard and no predictions were being shown, no feedback was received from the screen-reader. This misled users into believing that the screen reader had stopped working.

6. Conclusion and Future work

We designed an accessible version of a gesture and pop-up based keyboard *Swarachakra*, and compared its performance with Google Indic, a predominantly tap-based keyboard. We found that the gesture-based keyboard did significantly better than the tap-based keyboard. Even after 10 sessions and about 3 hours of typing practice on each keyboard, there seemed to be no evidence of saturation. Future research is needed with more sessions will be required to establish the saturation point. Although there are papers discussing the benchmark data for sighted users in Indic scripts, this is the first paper to provide benchmark speeds for text input by visually impaired in Indic scripts through an empirical study.

Our study also provides insights into future designs of Indic script keyboards. We attempted to help the users distinguish between short and long vowels by varying pitch and between similar sounding consonants by providing examples. Neither seemed to have helped enough, particularly as the users speed improved. Alternatives (such as bigger pitch differences or completely different voices) can be tried in future research. Gestures for space and backspace were not used much because users were concerned about the accidentally touching a key while executing the gesture, leading to unintended text input.

While ANOVA is often used to assess the difference between interventions in a within-subject design, we argue that for text input studies with novice users, a repeated measures ANOVA may not be the most appropriate test. When novice users learn to type using the first input method, they are learning to type in addition to learning to type using that specific input method. By the time they attempt their second input method, they are already at an advantage. This could result in a significantly better typing speeds on their second input method compared to their first one, which is not factored out by the repeated measures ANOVA. A multiple regression can tease out the actual differences between input methods after accounting for the differences between attempts.

7. Acknowledgements

This project was funded by Tata Consultancy Services. We thank the volunteers from XRCVC, Ruia College in Mumbai and National Association for Blind in New Delhi for participating in our studies and providing us with valuable feedback.

8. References

1. Srivastava M., Anu Bharath P,: Accessible Swarachakra: A virtual Keyboard for Visually Impaired. In: India HCI, 8th Indian Conference on Human Computer Interaction, pp. 111-115. ACM, New York (2016)
2. Joshi, Anirudha, Girish Dalvi, Manjiri Joshi, Prasad Rashinkar, and Aniket Sarangdhar.: Design and evaluation of Devanagari virtual keyboards for touchscreen mobile phones. In: 13th International Conference on Human Computer Interaction with Mobile Devices and Services, pp. 323--332 ACM, New York(2011)
3. Google Indic Playstore link, https://play.google.com/store/apps/details?id=com.google.android.apps.inputmethod.hindi&hl=en
4. Dalvi, G., Ahire, S., Emmadi, N., Joshi, M., Malsettar, N., Samanta, D., Jalihal, D., Joshi, A.: A Protocol to Evaluate Virtual Keyboards for Indian Languages. In: Proceedings of the 7th International Conference on HCI, pp. 27--38. ACM, New York (2015)
5. Bonner M.N., Brudvik J.T., Abowd G.D., Edwards W.K. (2010) No-Look Notes: Accessible Eyes-Free Multi-touch Text Entry. In: Floréen P., Krüger A., Spasojevic M. (eds) Pervasive Computing. Pervasive 2010. LNCS, vol 6030. Springer, Berlin, Heidelberg
6. Azenkot, S., Wobbrock, J., Prasain, S., Ladner, R. Input finger detection for nonvisual touch screen text entry in Perkinput. In. Proceedings of Graphics Interface, pp.121—129. ACM, New York 2012
7. MacKenzie, I. S., and Zhang, S. X. The design and evaluation of a high-performance soft keyboard. In: Proceedings of the SIGCHI conference on Human Factors in Computing Systems. pp. 25–31. ACM, New York (1999).
8. Kristensson, P. O. Discrete and Continuous Shape Writing for Text Entry and Control.

Doctoral dissertation, Linkoping University, (2007).
9. Kristensson, P. O., Vertanen, K.: The inviscid text entry rate and its application as a grand goal for mobile text entry. In: Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services, pp. 335—338. ACM, New York (2014).
10. Arrangement of Digits, Letters and Symbols on Telephones and Other Devices that can be Used for Gaining Access to a Telephone Network, http://www.itu.int/rec/T-REC-E.161-200102-I/en
11. CDAC InScript: Unified Virtual Keyboard for Indian Languages, 2015. http://www.cdac.in/index.aspx?id=dl_android_uvkil
12. *Sparsh* Hindi Keyboard, https://play.google.com/store/apps/details?id=com.sparsh.inputmethod.hindi&hl=en
13. SwiftKey Keyboard, 2015. https://play.google.com/store/apps/details?id=com.touchtype.swiftkey&hl=en
14. Swype Keyboard, http://www.swype.com/
15. Google Explore by touch, https://support.google.com/accessibility/android/answe r/6006598
16. Phraselist, https://docs.google.com/spreadsheets/d/1_2OZMz35WFgfpqQt2FSzBuwiPcVee546pZmyieXab_o/edit#gid=830720293
17. Vocaliser TTS Voice, https://play.google.com/store/apps/details?id=es.codefactory.vocalizertts&hl=en
18. Devanagri, https://en.wikipedia.org/wiki/Devanagari
19. Ghosh, S., Joshi, A., Joshi, M., Emmadi, N., Dalvi, G., Ahire, S., Rangale, S.,: Shift+Tap or Tap+LongPress?: The Upper Bound of Typing Speed on InScript. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, pp. 35-46. ACM, New York (2017)
20. E-Speak TTS, https://play.google.com/store/apps/details?id=com.googlecode.eyesfree.espeak&hl=en
21. Google TTS, https://play.google.com/store/apps/details?id=com.google.android.tts&hl=en
22. Rodrigues, A., Montague, K., Nicolau, H., Guerreiro, T.: Getting Smartphones to Talkback: Understanding the Smartphone Adoption Process of Blind Users. In: Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility, pp. 23-32. ACM, New York (2015)
23. Abugida, https://en.wikipedia.org/wiki/Abugida.
24. Estimation of blindness in India from 2000 through 2020: implications for the blindness control policy, https://www.ncbi.nlm.nih.gov/pubmed/11804362
25. Dalvi, G., Ahire, S., Emmadi, N., Joshi, M., Joshi, A., Ghosh, S., Ghone, P., Parmar, N.: Does Prediction Really Help in Marathi Text Input? Empirical Analysis of a Longitudinal Study. In: Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services, pp. 35—46. ACM, New York (2016)
26. Nicolau, H., Montague, K., Guerreiro, T., Rodrigues, A., Hanson, L.: Typing Performance of Blind Users: An Analysis of Touch Behaviors, Learning Effect, and In-Situ Usage. In:

Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility. pp. 273—280. ACM, New York (2015)