BrailleEasy: One-handed Braille Keyboard for Smartphones

Barbara ŠEPIĆ, Abdurrahman GHANEM and Stephan VOGEL
Qatar Computing Research Institute, Hamad Bin Khalifa University

Abstract. The evolution of mobile technology is moving at a very fast pace. Smartphones are currently considered a primary communication platform where people exchange voice calls, text messages and emails. The human-smartphone interaction, however, is generally optimized for sighted people through the use of visual cues on the touchscreen, e.g., typing text by tapping on a visual keyboard. Unfortunately, this interaction scheme renders smartphone technology largely inaccessible to visually impaired and blind people as it results in slow typing and higher error rates. Apple and some third party applications provide solutions specific to blind people which enables them to use Braille on smartphones. These applications usually require both hands for typing. However, Brailling with both hands while holding the phone is not very comfortable. Furthermore, two-handed Brailling is not possible on smartwatches, which will be used more pervasively in the future. Therefore, we develop a platform for one-handed Brailing consisting of a custom keyboard called **BrailleEasy** to input Arabic or English Braille codes within any application, and a **BrailleTutor** application for practicing. Our platform currently supports Braille grade 1, and will be extended to support contractions, spelling correction, and more languages. Preliminary analysis of user studies for blind Arabic-speaking participants showed that after less than two hours of practice, participants were able to type significantly faster with the BrailleEasy keyboard than with the standard QWERTY keyboard.

Keywords. Braille, keyboard, text entry, accessibility, smartphone, touchscreen, English, Arabic

1. Introduction

Braille is a famous writing system used by visually impaired people all over the world for nearly 200 years. The Braille cell is represented with 6 dots numbered from 1 to 6 (Figure 1). On papers, each of the dots can be embossed to be read by touch. Depending on the combination of raised dots, each cell represents one of 64 unique alphanumeric characters. Additional characters can be represented with two or more cells. A physical Braille keyboard usually consists of 6 main keys mapped to the 6 dots of a Braille character (left image in Figure 1) and extra keys for space, backspace and new line. To type a Braille character, the corresponding keys on the keyboard are pressed simultaneously. On touchscreen devices with multi-touch capabilities, many applications offer Braille support based on the same concept but replace the physical keys with virtual ones. However, screens on smartphones are often too small for 6 finger typing. Furthermore, this requires users to put their phone on a surface while typing. One-handed Brailling opens

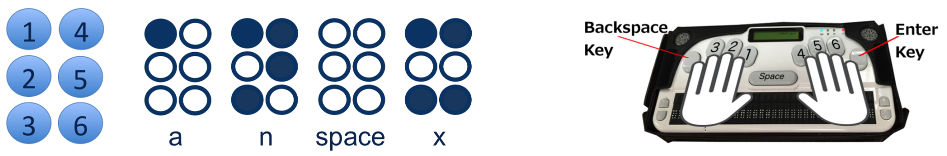

Figure 1. Left: Braille cell representation with example characters. Filled circles represent raised dots. Right: A typical Braille keyboard with 6 main keys – each mapped to a specific dot in a Braille character – and keys for space, backspace and enter.

new opportunities for the visually impaired to comfortably type with only one hand. The same input technique also works with smartwatches if multi-touch is supported.

Various solutions have been presented for eyes-free input. These solutions can possibly be categorized as non-Braille-based and Braille-based solutions. The former are mainly based on one of the standard keyboard layouts with audible feedback. Apple introduced the standard QWERTY keyboard with **VoiceOver**: a preinstalled accessibility feature that is widely used in the blind community. The user slides his finger over the keyboard, receives audio feedback for each character and selects the character by performing another gesture such as a double-tap or split-tap. **No-Look Notes** [1] outperforms the standard QWERTY keyboard with VoiceOver both in accuracy and speed. **NavTap** [2] is based on the T9 keyboard layout and shows slight performance improvements. However, such typing is quite slow as the user has to first locate the character and then perform another gesture to select it.

Since visually impaired users usually know Braille, many input solutions are based on Braille codes. With iOS 8.0, Apple introduced the **Braille Screen Input**, a two-handed Braille-based input screen that replaces the default keyboard within any application. However, it requires placing the device on a surface or holding it in an uncomfortable way, especially with the iPhone where it is available only in landscape mode. First attempts in two-handed Brailling (**BrailleTap** [2], **BrailleType** [3]) did not manage to outperform VoiceOver. On the other hand, **BrailleTouch** [4,5,6,7] is another widely used two-handed solution that adapts the principles of typical Braille keyboards, but requires that users hold their smartphone with screen facing away from them. **TypeInBraille** [8] and **Perkinput** [9] both introduce the concept of one-handed Brailling to overcome the usability issues of two-handed Brailling. The former divides the Braille character into three rows while the latter divides it into two columns. In both cases, the rows/columns are entered as sequential gestures. One-handed Brailling eliminates the usability issues as a user can hold the smartphone with one hand and use the other hand to type. However, all of the aforementioned third party solutions are standalone applications - the user can type the text in dedicated application but has to copy&paste or send it to the desired destination.

The main advantage of our work is that we offer an actual custom keyboard for one-handed Brailling (BrailleEasy) that can replace the standard keyboard to type *within any application*. Furthermore, BrailleEasy supports not only English, but also Arabic.

2. The One-handed Braille Keyboard

Our solution offers a complete platform that enables visually impaired users to comfortably type Braille codes with one hand. The platform consists of two components: The

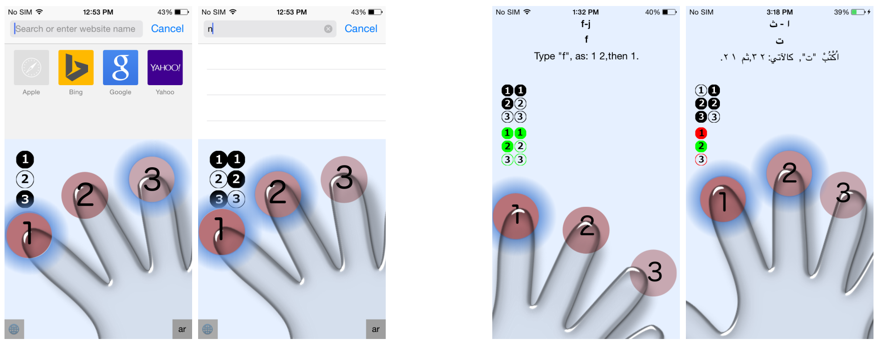

Figure 2. Left: Our BrailleEasy keyboard used to input a URL in Safari. User actions are accompanied with audio feedback. Right: One-handed BrailleTutor in English and Arabic audibly requests the user to type a Braille code and gives feedback after input via Siri, Apple's text-to-speech engine. Green circles indicate correct input while red circles indicate typing errors.

one-handed Braille keyboard, **BrailleEasy**, and the one-handed Braille tutorial application, **BrailleTutor** (Figure 2). BrailleEasy mimics the standard Braille Keyboard, but - similarly as in [9] - divides the typing of one character into two gestures as shown in Figure 2 and Figure 3. For example, to type *n* the user taps with the index and ring fingers to type the first column, then taps with the index and middle fingers to enter the second column of the Braille character. If a column is empty (none of three dots is risen), the user swipes across the screen with one finger.

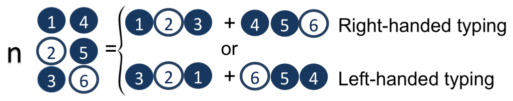

Figure 3. Each Braille character is divided into two gestures and can be oriented either for right or left hand.

Every user action is accompanied with audio feedback. For example, typing a character or word triggers reading it back to the user. Changing the keyboard audibly notifies the user which language is selected. Deleting a character or typing a new line are also accompanied with dictating the performed action.

Unlike e.g. BrailleTouch [5], finger positions are not restricted to fixed points on the screen nor to any specific orientation. That is, the user may move his hand while typing. We address this challenge by employing an adaptive calibration scheme in which 1) the user can reset calibration of finger positions at any time by long pressing with three fingers on the screen and 2) the calibration information is continuously updated at each user input. After each calibration reset, auditory feedback is given to notify the user. Recognition of three-finger tap is trivial. For any other finger combination, each tap has to be assigned to either the index, middle or ring finger by calculating the minimal distance to the updated calibration points.

Given that even proficient users of Braille may only be familiar with two-handed typing, some practice may be needed until users can comfortably type it single-handed. Therefore, we also developed the BrailleTutor application to serve as a tutorial for one-handed Brailling. The tutorial provides a set of lessons in both Arabic and English to help users practicing one-handed Brailling even if they have never learned Braille before.

3. Evaluation and User Studies

The user study is designed to measure typing speed on touch screen devices and the comfort of typing using different keyboards. The later can be roughly known by asking users to take a subjective questionnaire. The former can be measured by conducting a user study as described below.

Working closely with the Al Noor Institute for the Blind and the Qatar Social and Cultural Center for Blind (QSCCB[1]) we conducted a user study with 6 blind participants (2 female) aged between 19 and 50. All participants were proficient in reading and typing Braille but may not use it on daily basis. Although all of them were Arabic native speakers, 3 were highly proficient in English. We thus divided the participants in an Arabic and an English group.

The study consisted of four sessions: two practice and two testing (additional sessions are undergoing). The first two sessions were intended for users to get familiar with our tutorial, keyboard and software that was developed for this user study. In each session, they were asked to type with each of the keyboards for approximately 20min. The software read the short phrase they were requested to type, offered the spelling of the words and the possibility to listen to it as often as needed. Time was measured only after they started typing. The order of typing methods (keyboards) and sentences presented was randomly mixed across participants for counterbalancing. Between different methods, the users were requested to take a few minutes break. After finishing the first two practice sessions, the users had been typing for 20-60 minutes with BrailleEasy in total. The results below are based on measurements during the two following sessions.

The English phrases were adopted from the Children's Phrase Set that has previously been used to compare the typing speed of keyboards [10]. It consists of very short phrases that are easy to remember and do not include complicated words. Unfortunately, we have not found any equivalent set for Arabic. Therefore, we created a set of short phrases generated from Arabic subtitles of TED talks [11].

The most frequently used keyboard of choice for all participants was Apple's standard keyboard with VoiceOver (QWERTY) – i.e. all of them used QWERTY for typing on their phones for at least six months prior to testing. Therefore, we decided to compare our keyboard to QWERTY. Table 1 compares both keyboards for each participant separately. To have a comparison with other Braille keyboards, the users were asked to type in Braille via BrailleSense as well. BrailleSense is a very popular physical Braille keyboard (see Figure 1). Table 2 compares the typing speed on BrailleSense with our keyboard. However, as our primary aim was developing a software keyboard for handheld devices and not a replacement for a physical keyboard, we did not include BrailleSense in our further analysis. To compare the performance of QWERTY and BrailleEasy, English and Arabic data were analyzed separately to avoid conflating language effects with keyboard effects, as even participants in the English group were Arabic-language dominant.

For the current study, a 2x2 within-subjects factorial design study was carried out with two factors, Session (1, 2) and Method (BrailleSense, QWERTY). We measured the typing speed in characters per minute (cpm) and counted the number of corrections carried out while typing.

We analyzed the data using two 2x2 mixed model analyses of variance with fixed effects for Session and Method and random effects for Participant. To determine whether

[1]http://www.blind.gov.qa/

Table 1. Typing speed comparison across different devices: standard Apple's QWERTY keyboard with VoiceOver and BrailleEasy keyboard. Speed is measured in characters per minute (cpm).

(a) Typing speed [cpm] including mistyped characters.

	QWERTY	Braille-Easy
A1	**33.5**	25.2
A2	31.1	**43.7**
A3	28.6	**30.6**
E1	72.4	**83.2**
E2	36.4	**37.3**
E3	**28.2**	26.8

(b) Percentage of corrected errors (deleted characters).

	QWERTY	Braille-Easy
A1	1.8%	**14.2%**
A2	3.7%	**6.4%**
A3	1.1%	**3.4%**
E1	3.3%	**3.9%**
E2	2.0%	**3.6%**
E3	2.6%	**23.8%**

(c) Typing speed [cpm] excluding the corrected characters.

	QWERTY	Braille-Easy
A1	**32.2**	19.3
A2	28.9	**37.3**
A3	28.1	**28.2**
E1	67.5	**75.0**
E2	**34.7**	33.8
E3	**26.5**	17.3

Table 2. Typing speed [cpm] of BrailleEasy and physical Braille keyboard (BrailleSense). Factor specifies the magnitude by which the external keyboard is better.

	BrailleEasy	BrailleSense	Factor
A1	25.2	42.2	1.7
A2	43.7	110.6	2.5
A3	30.6	103.0	3.4
E1	83.2	162.8	2.0
E2	37.3	89.2	2.4

Table 3. Improvement in typing speed [cpm] with BrailleEasy keyboard over the two sessions.

	Session1	Session2	Improvement
A1	22.5	25.2	11.6%
A2	39.6	43.7	10.3%
A3	26.0	30.6	17.5%
E1	78.3	83.2	6.3%
E2	29.9	37.3	24.8%

counterbalancing had been effective, we also included Order in the model, and it was found to not be significant (Arabic data: $F_{3,183} = 0.18$, $p = 0.91$; English data: $F_{3,184} = 1.39$, $p = 0.24$).

Beginning with the English data, there was no significant interaction between Method and Session ($F_{1,184} = 0.56$, $p = 0.45$). There were main effects of both Method on speed in cpm ($F_{1,184} = 4.93$, $p < 0.05$) and Session on speed ($F_{1,184} = 7.79$, $p < 0.05$). Character-per-minute rate was higher for BrailleEasy ($M_{BE} = 57.18$) than for QWERTY ($M_Q = 45.69$). This shows that users typed on BrailleEasy significantly faster than on QWERTY. Furthermore, speed was higher for Session 2 ($M = 53.55$) than for Session 1 ($M = 49.32$) suggesting that the participants can improve their typing speed on BrailleEasy very quickly even with little training. The speed-up is shown in Table 3 for each participant.

In contrast, analysis of the Arabic data showed a significant interaction between Method and Session ($F_{1,183} = 4.82$, $p < 0.05$) which qualified the main effect of Session ($F_{1,183} = 11.63$, $p < 0.005$). In Session 1, QWERTY ($M_{Q,S_1} = 30.22$, $SD_{Q,S_1} = 2.69$) was faster than BrailleEasy ($M_{BE,S_1} = 29.41$, $SD_{BE,S_1} = 9.04$). However, in Session 2, BrailleEasy ($M_{BE,S_2} = 33.15$, $SD_{BE,S_2} = 9.54$) outperformed QWERTY ($M_{Q,S_2} = 32.08$, $SD_{Q,S_2} = 2.36$) for speed. This shows that although users started out typing faster using QWERTY, by Session 2, their typing was faster using BrailleEasy.

Many factors necessitate further investigation. Measurements from further sessions will help determine if BrailleEasy continues to grow in speed as participants learn. Furthermore, the low number of participants warrants continuing extension of the user study to additional participants as this will make the conclusions drawn here more robust.

After the last session, we asked the users to fill out a questionnaire about subjective preferences of keyboards in different situations. For long texts, most (4) preferred external keyboards. For short texts (e.g. chats, tweets, ...) users were divided between

QWERTY and BrailleEasy. However, the majority preferred BrailleEasy for situations like riding a bus, while walking or being at home (while lying or sitting on a sofa).

4. Conclusion and Future Work

BrailleEasy and BrailleTutor represent a platform that enables visually impaired and blind users to input text by typing Braille code within any application.

Current areas of development include extending the character set to support all special characters, capital letters and numbers, and offering better navigation possibilities. We expect that adding special characters will further improve the performance of BrailleEasy over QWERTY because QWERTY requires multiple interactions to enter them. Furthermore, we are planning to integrate a language model [12] to reduce gesture recognition errors. This will also allow us to do spell checking and correction, significantly enhance the gesture recognition algorithm and support Braille grade 2. Currently, our platform is designed to easily include more languages by simply mapping characters to the corresponding Braille representation.

References

[1] M. N. Bonner, J. T. Brudvik, G. D. Abowd, and W. K. Edwards. No-look notes: accessible eyes-free multi-touch text entry. In *Pervasive Computing*. Springer, 2010.

[2] T. Guerreiro, P. Lagoá, P. Santana, D. Gonçalves, and J. Jorge. NavTap and BrailleTap: non-visual texting interfaces. In *Resna*, 2008.

[3] J. Oliveira, T. Guerreiro, H. Nicolau, J. Jorge, and D. Gonçalves. BrailleType: unleashing Braille over touch screen mobile phones. In *HCI–INTERACT*. Springer, 2011.

[4] B. Frey, K. Rosier, C. Southern, and M. Romero. From texting app to Braille literacy. In *CHI'12 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2012.

[5] C. Southern, J. Clawson, B. Frey, G. Abowd, and M. Romero. An evaluation of BrailleTouch: mobile touchscreen text entry for the visually impaired. In *International Conference on Human-computer interaction with mobile devices and services*. ACM, 2012.

[6] B. Frey, C. Southern, and M. Romero. BrailleTouch: mobile texting for the visually impaired. In *Universal Access in Human-Computer Interaction. Context Diversity*. Springer, 2011.

[7] M. Romero, B. Frey, C. Southern, and G. D. Abowd. BrailleTouch: designing a mobile eyes-free soft keyboard. In *International Conference on Human Computer Interaction with Mobile Devices and Services*. ACM, 2011.

[8] S. Mascetti, C. Bernareggi, and M. Belotti. TypeInBraille: a braille-based typing application for touchscreen devices. In *International ACM SIGACCESS Conference on Computers and Accessibility*. 2011.

[9] S. Azenkot, J. O. Wobbrock, S. Prasain, and R. E. Ladner. Input finger detection for nonvisual touch screen text entry in Perkinput. In *Graphics Interface*, 2012.

[10] Kano, A., Read, J. C., Dix, A. Children's phrase set for text input method evaluations. In Proceedings of the 4th Nordic conference on Human-computer interaction. ACM, 2006.

[11] Cettolo, Mauro and Girardi, Christian. Wit3: Web inventory of transcribed and translated talks, In proceedings of the 16th Conference of the European Association for Machine Translation (EAMT) 2012

[12] X. Huang, A. Acero, H.-W. Hon, and R. Reddy. *Spoken language processing: A guide to theory, algorithm, and system development.* Prentice Hall PTR, 2001.