UNIVERSITÀ DEGLI STUDI DI
MILANO

Dipartimento di Informatica e Comunicazione

TypeInBraille: quick typing on smartphones by blind users

Rapporto interno N. 39 11 RT

Sergio Mascetti

Cristian Bernareggi

Matteo Belotti

TypeInBraille: quick typing on smartphones by blind users.

Sergio Mascetti
DICo - University of Milan
mascetti@dico.unimi.it

Cristian Bernareggi
DICo - University of Milan
bernareggi@dico.unimi.it

Matteo Belotti
University of Milan
matteo.belotti@studenti.unimi.it

ABSTRACT

Smartphones provide new exciting opportunities to visually impaired users because these devices can support new assistive technologies that cannot be deployed on desktops or laptops. Indeed, smartphones can be used in a number of situations in which it is not practical to rely on standard devices (i.e., desktop or laptops) and they are also equipped with sensors, like GPS receivers, accelerometers, and gyroscopes, that can be used to acquire information about the user's context.

Some devices, like the iPhone, are rapidly gaining popularity among the visually impaired[1] since the use of screen reader applications renders these devices accessible. However, there are still some operations that require a longer time or higher mental workload to be completed by a visually impaired user. In particular, in this paper we consider the problem of typing on devices not equipped with a physical keyboard. We propose a novel technique, called *TypeInBraille*, that is based on the Braille code and hence is specifically designed for blind users. The experimental evaluation we conducted demonstrates that our proposed solution achieves much higher performance when compared to the standard typing solution based on the on-screen QWERTY keyboard. In addition, *TypeInBraille* also enables the users to type when audio feedback is not reliable or even totally absent.

1. INTRODUCTION

The accessibility to smartphone devices by visually impaired users has recently significantly improved. This was obtained by adopting an interaction paradigm that couples the touch-screen with a speech synthesizer: in a nutshell, the visually impaired user can touch the screen to explore the interface, and can touch twice to activate an interface object (e.g., by touching an icon twice, the corresponding application is run). As a result of these improvements, most of the applications developed for sighted users can also be used by the visually impaired.

Despite these achievements, there are still some operations that require a longer time or higher mental workload to be completed by a visually impaired user. In particular, in this paper we consider the problem of typing. Since the large majority of smartphone devices do not have a physical keyboard, typing is enabled by the on-screen QWERTY keyboard that appears on the device screen. For a visually impaired user, inserting each character requires to search for the corresponding button on the keyboard. As reported in [6] and also highlighted in our experimental results, this operation is time consuming and error-prone. Intuitively, this is due to two main factors: first, the on-screen virtual keyboard contains about 30 buttons, hence the size of the buttons and the distance among them is necessarily small. Second, differently from the physical keyboard, there is no tactile feedback and the user must rely on the audio feedback only.

To address this problem, in this paper we propose *TypeInBraille*, a novel approach to text typing that is specifically designed for blind users. The core idea is to allow typing using the Braille code, which is generally employed for reading by blind people. More specifically, we propose a technique that decomposes the 6-dots Braille cell representing a character into three 2-dots sub-symbols. Each of the four possible sub-symbols is associated with a gesture (i.e., a touch on the screen). Consequently, the user can type a character with a sequence of three gestures or less (as we describe in the following, in many cases the user can use just one or two gestures).

In this paper we describe the implementation of this idea into an iPhone prototype that supports a blind user in the process of typing and editing, and that provides the functionalities to copy the resulting text into the clipboard or to send it via email or text message.

The proposed solution has a number of advantages with respect to the on-screen keyboard and the other typing techniques designed for blind users. In this paper, we experimentally show that:

- Typing with *TypeInBraille* requires less time with respect to the on-screen keyboard and is less error-prone.

[1] `http://behindthecurtain.us/2010/06/12/my-first-week-with-the-iphone/`

- Learning to use *TypeInBraille* takes only a few minutes by people that already know the Braille code.

- Differently from what happens with the on-screen keyboard, when using *TypeInBraille* the typing performance is not significantly affected by an uncomfortable environment in which the audio-feedback is unreliable or even absent and the user is subject to sudden movements (e.g. when the user is traveling on a tramcar).

The paper is organized as follows. In Section 2 we summarize the existing solutions to the problem of typing on smartphones by visually impaired users, highlighting the main differences with respect to our approach. In Section 3 we identify the problems related to typing on a touchscreen device for the visually impaired, focusing on the problems related to on-screen keyboards. In Sections 4 we describe in details our solution while in Section 5 we describe our prototype implementation and report and discuss the results of our experimental evaluation. Finally, Section 6 concludes the paper and identifies future work.

2. RELATED WORK

Text entry on mobile devices has been a challenging issue for a long time. The small physical dimensions of these devices limit the number and the size of the available keys (both physical or on-screen keys). Therefore, many text entry strategies have been studied for years to strike a balance among typing speed, accuracy and the keyboard size.

All of these text entry solutions have posed accessibility and usability problems to many categories of users and especially to blind users, who cannot type and check at the same time what is being dynamically displayed on the screen. Most of these issues have been addressed by designing text entry techniques which couple multi-tapping (i.e., repeatedly pressing the same key to obtain a character) and speech output. For example, the "Talks" application implements this paradigm on Symbian phones. However, studies on the typing speed by blind users have proved that multi-tapping is a rather slow input method [6].

In order to increase typing speed with devices equipped with a physical keyboard, an alternative solution was proposed by Guerreiro et al. with the "NavTap" system [3]. The "NavTap" text entry system displays letters in five rows and alphabetically ordered. Each row starts with a vowel so that at most six letters are displayed in a row. Navigation among letters is enabled by four physical keys. Tests conducted with 10 blind users unfamiliar with mobile devices show that the maximal writing speed that can be achieved is around 8.5 words per minute (wpm).

A different approach consists in adding a specialized keyboard to the mobile device. The application called "RearType" [7] enables the blind user to write on a keyboard placed on the back of the device. A test with 7 users proved that the highest typing speed is around 15 wpm. Another solution based on a specialized keyboard is "Twiddler" [5] that relies on a keyboard with 12 keys to be used with one hand. A test with 10 blind participants highlighted that the average typing speed is around 19.8 wpm.

In this paper we consider the devices that do not have a specialized keyboard for blind users nor a general purpose physical one. Clearly, with these devices, both typing speed and accuracy decrease with respect to typing on a device with a physical keyboard. A promising solution developed specifically for the blind people is "NavTouch" [2], an improvement of "NavTap" that enables blind users to type with the touchscreen on a smartphone. The character layout is analogous to the one used in the "NavTap" solution; the main difference is that, instead of the physical keys, "NavTouch" adopts directional gestures on the touchscreen to navigate and find the target character. Special actions are associated to screen corners, since they can be easily identified by touching the side of the smartphone (e.g. bottom-right corner is used to delete the last character). The experiments presented in [2] show that the number of times users need to re-enter a character due to a wrong typing is between 12% and 25% while, with our solution (see Section 5), this value is about 8%. For what concerns typing performances, no data is available to directly compare *TypeInBraille* with "NavTouch". An indirect measure is given by the key strokes per character (kspc); with "NavTouch" the theoretical limit of this metrics is 2.75 kspc (considering the average for all the letters in the alphabet), while with *TypeInBraille* this value is slightly lower than 3. However, while with "NavTouch" the actual number of kspc measured with unexperienced users is about 5, with our technique this value is constant (i.e., less than 3).

A different text entry technique designed to improve typing speed by blind users was proposed in the "Adaptive Blind Interaction" method [8]. It is based on a three-layer pie menu. Each layer makes available eight letters which can be reached by sliding the finger along eight directions. The layer adaptively changes after a certain time slot. One out of three blind users involved in a usability test was able to achieve a typing performance equals to 12 wpm after a long training (about 13 hours). Nonetheless, no quantitative data is available about average performance and the user profile. These experimental results seem to be in contrast with the results obtained with a similar technique called "No-look notes" [1]. This solution is similar to the one proposed in [8] as it is based on a two-layer pie menu presenting letters arranged like in a numeric keypad. However, in this case, experimental results highlight that the three users involved in the tests achieved very low typing performances (1.32 wpm on average) during a one-hour long test. As we show in Section 5, both the iPhone on-screen keyboard and our solution enable the blind users to write much faster than 1.32 wpm (about 3 and 4 time faster, respectively). In addition, our solution can be effectively used after only a few minutes of training.

3. PROBLEM ANALYSIS

Over the last three years the iOS-based devices (which include the iPhone, iPod touch and iPad) have been getting more and more widespread among blind people, thanks to the full-featured multilingual screen reader, known as VoiceOver, natively embedded in the iOS system. In this section we analyze the hindrances that a blind person can meet while typing and editing text on an iPhone or iPod

touch[2]. It is worth noting that the analysis reported in this section also applies to Android devices, since the eyes-free text entry technique made available by VoiceOver has been recently adopted also by TalkBack, the most widely used screen reader for the Android system.

Section 3.1 introduces typing and editing techniques for iPhone with VoiceOver, while Section 3.2 describes the hindrances for the blind users showing that different problems can arise in different scenarios.

3.1 Text-entry on iPhone

The iPhone text entry technique is based on the spatial position of keys on the touchscreen. The iPhone on-screen keyboard arranges keys in three layers. In the first layer a QWERTY layout is displayed, whereas in the second and in the third layer are displayed numbers, punctuation marks and other special characters (e.g., the "+" or "#" symbols). A key is available for changing layer. Figure 1 shows the three layers of the on-screen keyboard; the keys used to move among the layers are circled in white.

(a) Layer1 - QWERTY (b) Layer2 - Numbers and symbols

(c) Layer3 - Other symbols

Figure 1: The three layers of the on-screen keyboard

The keyboard can be accessed by a blind user via VoiceOver. The input of a key is divided into two stages: *identification* of the target key and its *confirmation*. In the identification stage, the blind user scans the keyboard searching for the target key while the names of the touched keys are read by speech as soon as the keys are touched. For what concerns the confirmation, two modes are available: *keypad* mode and *touch* mode. In keypad mode, the confirmation is performed by tapping with another finger on the touchscreen. In touch mode, a key is confirmed when the user picks up the finger. In both modes, control and editing keys (e.g. layer change, delete, return, etc.) are used similarly to the keypad mode i.e., tapping with another finger.

A special input mode is available for inserting letters with accents (e.g., " è ") or other modifiers (e.g., " ñ " or " ö "). The user can type these letters with two gestures. First, a keyboard layer with accented or modified letters can be opened by double tapping and holding the finger down on the letter (see Figure 2). Then, the target accent or modifier can be selected by sliding the finger over the display and confirming the key. Note that these letters are hard to insert in touch mode because whenever the finger is picked up to achieve the double tapping gesture, the corresponding letter is entered without the modifier or the accent.

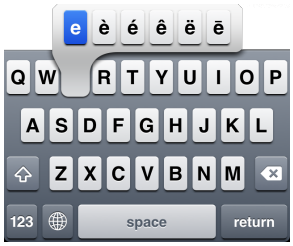


Figure 2: Inserting letters with accents and other modifiers.

Editing operations can be performed as follows. The user can move the cursor left or right by flicking the finger up or down, respectively, on the screen. By default, the cursor moves to the previous or next character. However, by using the "rotor", a gesture consisting in rotating two fingers on the touchscreen, the user can express the preference to move word-by-word or line-by-line. The rotor can also be used to select the word where the cursor is located, and to perform editing operations i.e., copy, cut and paste.

3.2 Hindrances for blind users

The text entry technique made available by VoiceOver on iPhone turns out to be usable in many scenarios and by many categories of vision impaired users. Nonetheless, in some scenarios, totally blind users meet some problems which significantly affect typing performance and accuracy. In the following we describe five main problems that we identified during the experiments we conducted with blind users. We distinguish two scenarios: when typing and editing is performed in a comfortable environment (e.g., the user is sitting at the desk) or in a setting in which the audio feedback is unreliable and the user is subject to sudden movements (e.g., while the user is standing on a tramcar).

The **first** issue a blind user runs into while typing on iPhone deals with the identification of the target key. Indeed, many keys are displayed in a single keyboard layer (up to ten keys per row arranged in four rows). Hence a single key has a small size (approximately $4x6$mm in portrait mode). By sight, even a small target key can be pointed out on the keyboard quickly and accurately. Instead, a blind user has to explore the keyboard with the fingertip before finding the right target key. Moreover, since on-screen keys are not perceptible by touch, there is no way for the user to instantaneously locate a target key among the others under the fingertips. Due to the small size of keys and to the lack of tactile feedback, also a skilled blind user can rarely identify a key without first exploring the part of the keyboard around it. This issue is exacerbated in an uncomfortable environment. For example, a sudden movement occurring while searching for a key may change the position of the fingertip on the touchscreen, hence forcing the user to restart

[2]For the sake of brevity, in the following we refer to "iPhone" meaning "iPhone or iPod touch".

the process. Note that, while this hindrance clearly affects typing performance, it also affects accuracy because in the search process it also happens to identify the wrong key especially in noisy environments in which the audio feedback less reliable.

The **second** issue concerns the target key confirmation stage. Indeed, due to the small size of keys, it is relatively frequent that the user trying to enter a key inadvertently slide the finger over another close-by key, thus causing a typing mistake. Moreover, in an uncomfortable environment, especially when using the touch mode, a key can be typed by mistake because of a sudden movement that makes the finger pick up on the wrong key. This issue affects typing accuracy as the user may not realize the typing mistake, but it also affects efficiency because when the user is aware of the mistake he/she has to correct it by deleting the character and re-entering the correct one.

The **third** issue, mainly affecting the typing performance, deals with the key arrangement in layers. A target key, not available in the current layer, must be typed in two stages: by first setting the target layer and then by identifying and confirming the desired key. Setting the target layer is an additional operation that requires the user to find out and confirm the changing layer key. In some cases, this operation needs to be performed twice (i.e., to move from Layer 1 to Layer 3, see Figure 1). Moreover, the time needed to identify an infrequently used target key (like the "+" symbol, for example) is much higher than the time required for a frequently used key because the user may not remember neither which is the layer containing the key nor its position within the layer.

The **fourth** problem concerns the keys that are used more frequently like, for example, the "blank space" and the "delete". The problem is that no shortcut exists to enter these keys more quickly than the other less-frequently used keys. On one side, this solution makes the use of the system easier as it does not introduce special gestures; however, on the other side, it also prevents the skilled users from achieving high typing performance.

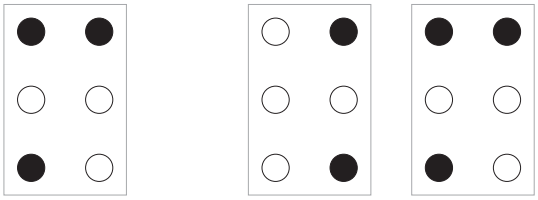
The **fifth** and last issue is that the text-entry technique via VoiceOver is fully dependent on speech output. Actually, without the speech feedback it would be impossible for the blind user to locate the target key, and to understand which character is inserted and even which layer is selected. Consequently, typing and editing text turn out to be difficult tasks in noisy environments (e.g., a tramcar) as well as in those cases in which the user cannot be distracted by the speech output (e.g., while walking aided by a white cane).

4. THE PROPOSED SOLUTION

This section describes *TypeInBraille*, the text-entry technique designed to overcome the issues highlighted in Section 3. Based on the problem analysis, a text entry technique addressed especially to blind users should satisfy the following requirements: it should allow quick typing and editing, it should not be error-prone, it should be usable also in the scenarios in which the user is subject to sudden movements and the audio feedback is unreliable or totally absent. In order to satisfy these requirements, we designed a text entry technique based on the Braille reading and writing system, especially known by blind people. Section 4.1 briefly introduces the Braille code while Section 4.2 illustrates the *TypeInBraille* typing technique and Section 4.3 highlights the advantages of our solution.

4.1 The Braille system

Braille is a text coding designed to be read by blind users through fingertips. Each character is represented by a cell made up of six dots organized into three rows of two dots each[3]. Each dot may be raised, hence perceptible with the fingertips, or flat. Consequently, each cell can represent up to 64 different characters. This value is sufficient for representing the lower-case letters (see, for example, the letter "m" reported in Figure 3(a)), the punctuation marks and some special characters (e.g., "@" and "&"). Some symbols, like the upper-case letters and the numbers, are represented by means of a prefix. For example, capital letters (Figure 3(b) reports an upper-case "m") are represented by means of the capital letter symbol, followed by the letter itself.



(a) Lower-case "m" (b) Upper-case "m"

Figure 3: Examples of Braille encoding (the black circle stands for a raised dot, while the white circle stands for a flat one).

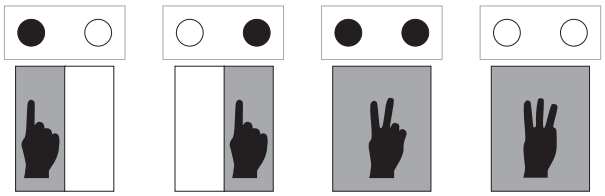
The Braille system is not a world standard. Indeed, there exist many national standards for Braille. Since we performed tests with Italian users, the application we implemented employs the official Italian Braille code approved as standard in 1998 [4]. However, the representation of the letters and numbers is generally very similar among the different national standards (for example, all the lower-case letters in the Italian standard are the same as the letters in the French, English and German standards). Consequently, we have no reasons to believe that results different from those reported in this contribution would be obtained with a different Braille standard.

4.2 The *TypeInBraille* typing technique

The proposed typing technique enables the user to input a character through its Braille representation by inserting the three rows of each cell from the top to the bottom. In order to enter a row (we recall that a row is composed by two dots), the touchscreen is divided into two rectangles (left and right) and four gestures are defined. A tap on the left part of the screen corresponds to the left dot raised and the right dot flat (Figure 4(a)). Similarly, a tap on the right part corresponds to the right dot raised and a left dot flat

[3]Although 8-dots representations are used as well, the 6-dots coding is the only Braille standard in many countries.

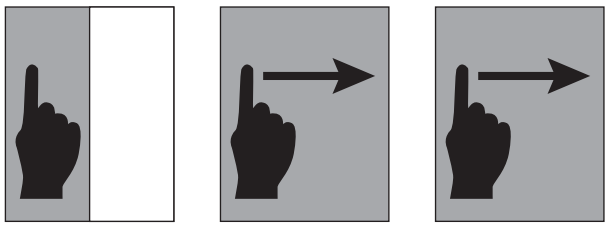
(Figure 4(b)). A double tap (i.e., a tap with two fingers) represents two raised dots (Figure 4(c)) while a triple tap stands for two flat dots (Figure 4(d)). After a character is entered, it is read by speech to the user and/or a vibration effect can be triggered.



(a) Left dot (b) Right dot (c) Both dots (d) No dot

Figure 4: The four gestures defined to enter a pair of dots.

In order to insert a Braille symbol three gestures would be required with our technique. However, we introduced an optimization that makes it possible to represent three frequent characters (blank space, "a" and "c" whose frequency is about 1/3 in Italian and English) with less gestures. In more details, an additional gesture was introduced to represent the end of a character i.e., all the following rows contain flat dots only. This gesture is the "one finger right flick", a movement with one finger from left to right. Since the letters "a" and "c" have raised dots on the first row only, they can be represented with two gestures, while the blank space, which contains no raised dots, can be represented by one gesture only (i.e., the one finger right flick). For example, as shown in Figure 5, the letter "a" followed by a blank space can be entered by the left dot gesture followed by two end of character gestures, the former indicating the end of the "a" character, the latter representing the blank space.



(a) Left dot (b) End of charater (c) Blank space

Figure 5: Letter "a" followed by a blank space.

Since the flick gesture is easy to remember and quick to perform, our technique adopts it also to insert a new line (one finger down flick) and to perform undo/delete operations (one finger left flick). More specifically, the latter gesture undoes the character that is currently being inserted or deletes the character on the left of the cursor in case the user is not in the process of inserting a character.

Text editing can be achieved thanks to three working modes, which can be set through the rotor gesture: "writing mode", "exploration mode" and "selection mode". In "writing mode", the user can type text as explained above as well as perform deletion operations. In "exploration mode" and "selection mode" the user can move the cursor through the inserted text. Additionally, in "selection mode", while the user moves the cursor, the text is also selected. These two modes share the same gestures. One finger left (or right) flick moves the cursor one character left (or right, respectively). The same gestures performed with two fingers move the cursor word-by-word while, using three fingers, the cursor is moved to the beginning or to the end of the text (three fingers left flick or three fingers right flick, respectively).

4.3 Advantages of our technique

TypeInBraille overcomes the issues analyzed in Section 3. The first issue, concerning the identification of the target key is addressed by leveraging the user from exploring the touchscreen to identify a target key. Indeed, with our technique, only two wide rectangles (i.e., the left or the right ones) must be identifiable on the screen and the user is able to find them instantaneously thanks to the tactile feedback given by the physical borders of the device. Hence, the target rectangle identification process is performed in a negligible time also by unexperienced users.

TypeInBraille addresses the second problem, namely the target key confirmation, by not requiring a confirmation at all. Indeed, with our solution, the user knows in advance the sequence of gestures needed to type a character and these gestures can hardly be confused among themselves. Consequently, while typing a character, there is no need to receive feedbacks from the device and hence there is no need for a confirmation by the user.

As for the problems arising from the use of layers, thanks to the typing strategy based on Braille, the user is enabled to type up to 64 characters in the same layer, including all the letters, even modified ones (e.g., accented vowels), punctuation and some special characters such as parentheses and arithmetic operators. Hence, typing performance increases, since no additional operation to set the right layer is needed. On the contrary, numbers and capital letters can be typed in two stages. A capital letter can be typed by first inserting a prefix (i.e., a capital letter sign) followed by the letter itself. Numbers can be typed by first inserting a prefix (i.e., a number sign) followed by the digits (each digit is one of the letters from "a" to "j"). Consequently, for numbers and capital letters, both the on-screen keyboard and *TypeInBraille* require an additional operation (i.e., the layer switch in the former case, the prefix in the latter one). However, differently from what happens with the on-screen keyboard, most infrequent characters can be typed in no more time than average used characters because the user knows their representation by heart.

Our technique was also optimized for frequently used keys. As remarked above, using *TypeInBraille*, most frequently used keys can be typed through one or at most two gestures (i.e., "a", "c", blank space and new line). Furthermore, also the gesture for deleting one character, which may be very frequently used in uncomfortable environments, was designed to be performed straightforwardly. This gesture not only improves accuracy, since the user can delete a character more easily than with the on-screen keyboard, but it also improves efficiency, because the time needed to correct a typing mistake is lower.

For what concerns the fifth problem identified in Section 3, it is worth remarking that, unlike typing on the on-screen keyboard supported by speech synthesizer, *TypeInBraille* turns out to be partially or even totally independent on speech output. Actually, as we experimentally show in Section 5, the audio feedback is not indispensable, because the characters are entered through gestures that are almost position independent. In the scenarios in which the audio feedback is not reliable or absent, the vibration effect alerts the user whenever a character is typed.

5. EXPERIMENTAL RESULTS

In this section we discuss the results of the experimental evaluation we conducted with blind users. In Section 5.1 we introduce the prototype implementation we used in our test, in Section 5.2 we describe how we conducted the experiments and finally, in Section 5.3 we report the results of our evaluation.

5.1 The Prototype implementation

In order to assess *TypeInBraille* with blind users, we implemented the technique in a prototype application. Since iPhone and iPod are getting widespread among blind people, especially thanks to the natively available screen reader VoiceOver, we chose to implement the prototype for iOS systems.

The prototype is a text editor that can be used both to type and edit text and that allows to change the application mode (from "writing mode" to "exploration mode" and to "selection mode") through the use of the rotor. Furthermore, the prototype enables the user to copy the text into the clipboard or to send it as an email or as a text message.

A peculiar implementation choice concerns the speech output. Although VoiceOver was the first choice for vocal reading, it turned out to be not usable. This is due to the fact that, when VoiceOver is active, most touch gestures (like single, double and triple tap) are handled by the operating system and they cannot be caught at application level. Consequently, the prototype must be used when VoiceOver is turned off. In this case, the speech output is obtained by invoking some of the "VoiceServices" system calls that are available in some versions of the iOS systems but that are not currently officially released by Apple.

5.2 Experimental setting

The aim of the experimental evaluation was to assess the effectiveness and efficiency of the *TypeInBraille* typing technique compared to writing with the on-screen QWERTY keyboard available on the iPhone mobile device.

The experimental evaluation was conducted with a group of seven users with the following characteristics.

- Congenitally blind. We focused our attention only on congenitally blind users in order to have a uniform sample.
- Skilled in reading 6-dot literary Braille.
- Skilled in writing with an iPhone on-screen QWERTY keyboard assisted by speech output via VoiceOver.
- Never used *TypeInBraille* before.

Before the evaluation, the users were trained for less than 5 minutes to use *TypeInBraille* and had 15 minutes to get familiar with the typing gestures. During the evaluation, the users were required to write a sentence in Italian that they learned by heart before the beginning of the experiment. This sentence is 111 characters long, consists in 20 words and contains punctuation marks, capital letters and apostrophes.

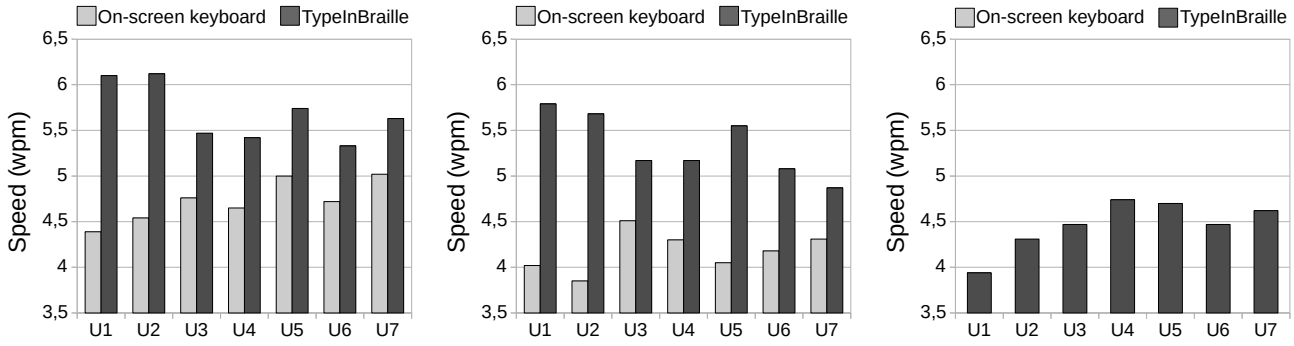
The experiments took place in three different scenarios. In the "desk" scenario, users were sitting at a desk and could use the iPhone with both hands. Users were required to write the sentence using the on-screen keyboard and *TypeInBraille*. In the second scenario, called "tramcar", users were asked to write the sentence on the move, using his/her preferred hand, standing in a noisy tramcar and keeping the iPhone in his/her pocket. Due to the noisy environment, the speech output was only partially audible. The last scenario took place in the same situation of the "tramcar" scenario. However, in this case, users wrote the sentence using *TypeInBraille* without any speech feedback. For this reason we call it the "no-audio" scenario. The only aid for the user was the vibration effect triggered by *TypeInBraille* as soon as a Braille symbol was inserted. In this scenario the users were not asked to write with the on-screen keyboard as this is not feasible for a blind user.

In order to assess the efficiency of both the on-screen QWERTY keyboard and *TypeInBraille*, four indicators were measured: total typing time, characters per minute and words per minute. Concerning the evaluation of the effectiveness of both techniques, two indicators were measured: number of errors and understandability. The former refers to the number of incorrect characters that each user inserts and does not correct. More precisely, with "incorrect characters" we intend missing ones, substituted ones or added-by-mistake ones. The latter indicator was expressed as the number of words understandable by a reader who has not ever read the sentence. Among the other parameters we measured, in the following we also report about the number of delete/undo operations as we believe that this metrics can raise interesting considerations.

5.3 Experimental results

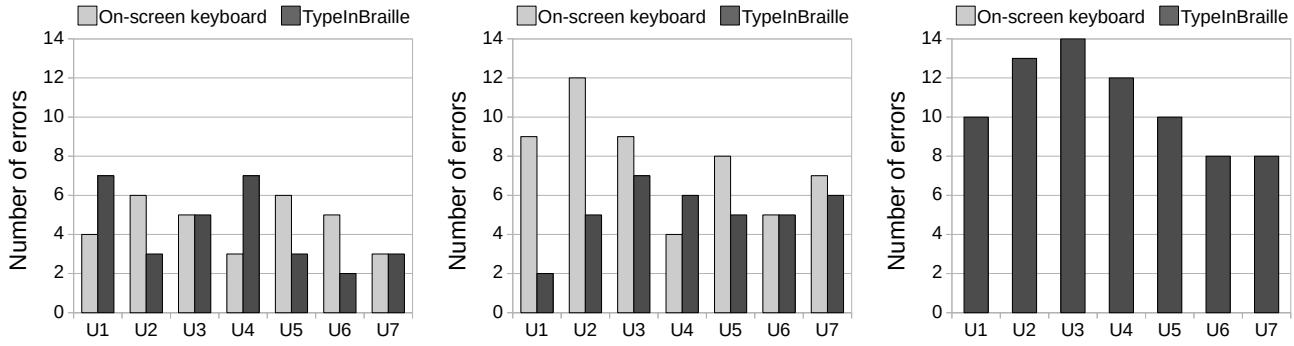
In the "desk" scenario, *TypeInBraille* increases the average typing performance up to about 17% with respect to the on-screen QWERTY keyboard. In more details, as shown in Figure 6(a), the typing performance of all the users is improved when using *TypeInBraille* and the average time to write the sentence is about 254 seconds (about 5.2 wpm) with the on-screen QWERTY keyboard and about 211 seconds (about 6.3 wpm) with *TypeInBraille*.

For what concerns the number of errors, one user performed better with the on-screen QWERTY, two users obtained the same result with the two techniques while four users improved their typing accuracy. In particular, the total number of errors for all the users with the on-screen QWERTY keyboard is 32 (corresponding to about 4% of wrong characters), while it is 25 with *TypeInBraille* (corresponding to about 3% of wrong characters). See Figure 7(b) for the de-



(a) "Desk" scenario. (b) "Tramcar" scenario. (c) "No-audio" scenario.

Figure 6: Evaluation of the typing speed (in words per minute).



(a) "Desk" scenario. (b) "Tramcar" scenario. (c) "No-audio" scenario.

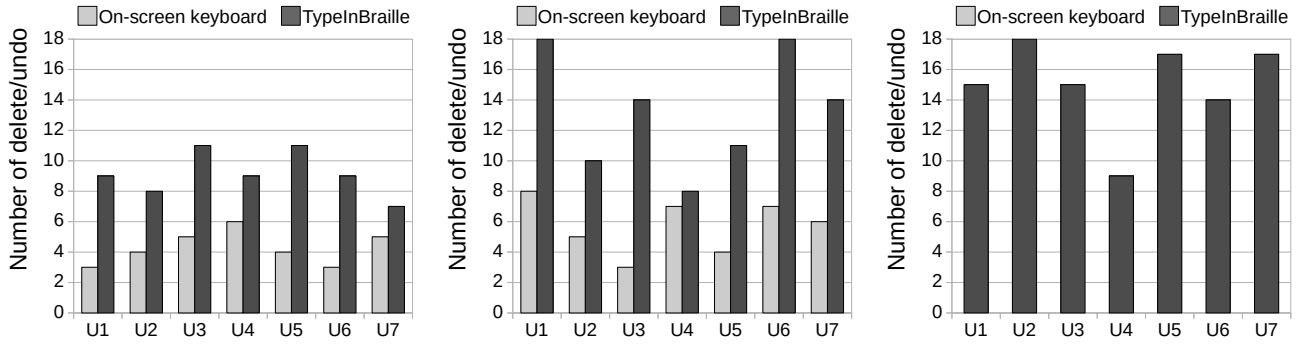
Figure 7: Evaluation of the number of errors.

tailed number of errors for each user. Consequently, *TypeInBraille* reduces the number of errors by 22%.

In the "tramcar" scenario the advantages of *TypeInBraille* are even more remarkable. Indeed, *TypeInBraille* improves the average typing performance by about 21% (287 seconds vs. 226 seconds). The benefits of *TypeInBraille* in this scenario are more significant than in the "desk" scenario because the time needed to write the sentence on the tramcar with the on-screen QWERTY keyboard is about 13% higher than in the "desk" scenario. Vice versa, the impact of the uncomfortable environment on *TypeInBraille* is less significant; indeed, in the "tramcar" scenario the time needed to write the sentence is 7% higher than in the "desk" scenario. This difference can also be observed by comparing Figures 6(a) and 6(b). A similar reasoning applies to the number of errors (see Figures 7(b)). Indeed, in the "tramcar" scenario the number of total errors obtained with *TypeInBraille* is reduced by about 1/4 with respect to the on-screen QWERTY keyboard (from 54 total errors with the on-screen QWERTY keyboard to 41 total errors with *TypeInBraille*). In this case one user performed better with the on-screen QWERTY keyboard, one user obtained the same result with the two techniques, while 5 users improved their typing accuracy.

The results in the "no-audio" scenario give evidence that *TypeInBraille* can also be used in the extreme case in which no audio feedback is available. Indeed, as can be observed in Figure 6(c), the typing performances are degraded but they are still better than those obtained with the on-screen keyboard in the "tramcar" scenario. As expected, the number of errors is significantly higher than in the other two scenarios as the user cannot understand when he/she enters a wrong character (see Figure 7(c)). Indeed, the total number of errors is 75, which is about 83% higher than in the "tramcar" scenario. However, it should be observed that the word understandability ratio is still above 95% (this metrics is about 100% in the other scenarios). Actually, most of the typing mistakes in the "no-audio" scenario are blank characters. This is due to the fact that the same gesture is used to insert a blank character and to confirm the end of a character being input. Without the speech feedback, it happens that the user is not sure about the termination of a character, hence he/she makes the right flick gesture that can result in the wrong insertion of a white space. Nonetheless, generally, one or more blank characters do not affect the understandability of a word.

It is remarkable that the number of delete/undo operations is approximately doubled with *TypeInBraille* than with the on-screen keyboard (see Figure 8). On one side, this measurement, coupled with the increasing performance and accuracy, clearly indicates that with *TypeInBraille* the gesture for delete/undo operations can be used very easily. However, on the other side, this result suggests that, although



(a) "Desk" scenario. (b) "Tramcar" scenario. (c) "No-audio" scenario.

Figure 8: Evaluation of the number of delete/undo.

the overall performances increase, the users experience some difficulties while using *TypeInBraille*. Thanks to the feedback provided by users, we suppose that the main problem is related to the fact that blind users memorize each Braille cell as a single element and its mental decomposition into the three sub-symbols demands a high cognitive workload. However, we recall that the users involved in our experiments practiced with *TypeInBraille* for a few minutes only so they had no time to practice with this problem. Based on this reasoning we believe that after a longer training users could get familiar with this mental exercise hence incurring in less mistakes and consequently obtaining even better typing performance.

6. CONCLUSIONS AND FUTURE WORK

In this paper we illustrate *TypeInBraille*, a novel typing technique that allows blind persons to write using a smartphone touchscreen. In our analysis, conducted with the help of blind users, we identify the main hindrances that arise while typing on a smartphone that is not equipped with a physical keyboard. The solution we propose alleviates or totally solves most of these problems. This statement is supported by our quantitative experimental analysis that shows how the typing speed and accuracy are improved by *TypeInBraille* with respect to the standard on-screen keyboard. This result is particularly significant because the users involved in the experiments were all experienced with the on-screen keyboard while they were only trained with *TypeInBraille* for less than 20 minutes. Additionally, our solution has benefits other than the writing performances; for example, it can also be used in the extreme case in which the user cannot be aided by any audio feedback.

For what concerns the future work, we intend to focus our effort in three different directions: the technique, the experimental analysis and the implementation. For what concerns the technique, one feedback we received during the tests highlighted that our solution requires the use of at least three fingers. Consequently, it is not possible to use *TypeInBraille* holding the device in one hand and typing with the thumb. We intend to investigate how to re-define the gestures so that all of them can be performed using the thumb only. Regarding the experimental analysis, we plan to extend it to a larger set of users and, in particular, to evaluate how the performance improves when the users get more experienced. Finally, we would like to port the application on the Android platform, possibly inserting the solution at operating system level so that it could be used as the typing technique for every other application in the system.

7. REFERENCES

[1] Matthew N. Bonner, Jeremy T. Brudvik, Gregory D. Abowd, and W. Keith Edwards. No-look notes: Accessible eyes-free multi-touch text entry. In Mirjana Spasojevic Patrik Floréen, Antonio Krüger, editor, *Pervasive*, volume 6030 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2010.

[2] Tiago Guerreiro, Paulo Lagoá, Hugo Nicolau, Pedro Santana, and Joaquim Jorge. Mobile text-entry models for people with disabilities. In *Proc. of the 15th European conference on Cognitive ergonomics*, ECCE '08, pages 39:1–39:4. ACM, 2008.

[3] Tiago Guerreiro, Hugo Nicolau, Joaquim Jorge, and Daniel Goncalves. Navtap: a long term study with excluded blind users. In *Proc. of the 11th int. ACM SIGACCESS conference on Computers and accessibility*, Assets '09, pages 99–106. ACM, 2009.

[4] Commissione italiana per il Braille. *Codice Braille italiano*. Biblioteca Italiana per Ciechi Regina Margherita, Monza, 1998.

[5] Kent Lyons, Thad Starner, and Brian Gane. Experimental evaluations of the twiddler one-handed chording mobile keyboard. *Human-Computer Interaction*, 21:343–392, 2006.

[6] Hugo Nicolau, Tiago Guerreiro, Joaquim Jorge, and Daniel Gonçalves. Proficient blind users and mobile text-entry. In *Proc. of the 28th Annual European Conference on Cognitive Ergonomics*, ECCE '10, pages 19–22, New York, NY, USA, 2010. ACM.

[7] James Scott, Shahram Izadi, Leila Sadat Rezai, Dominika Ruszkowski, Xiaojun Bi, and Ravin Balakrishnan. Reartype: text entry using keys on the back of a device. In *Proc. of the 12th int. conference on Human computer interaction with mobile devices and services*, MobileHCI '10, pages 171–180. ACM, 2010.

[8] Georgios Yfantidis and Grigori E. Evreinov. Adaptive blind interaction technique for touchscreens. *Universal Access in the Information Society*, pages 328–337, 2006.