**TypeInBraille**: A Braille-based Typing Application for Touchscreen Devices.

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. Some devices, like the iPhone, are rapidly gaining popularity among the visually impaired since the use of pre-installed screenreader 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 this contribution we present a novel application for text entry, called TypeInBraille, that is based on the Braille code and hence is specifically designed for blind users.

**Categories and Subject Descriptors**

H.5.2 [Information Interfaces and Presentation]: User Interfaces—Haptic I/O; H.5.2 [Information Interfaces and Presentation]: User Interfaces—Input devices and strategies

**General Terms**

Human Factors, Algorithms.

**Keywords**

Blind, Braille, QWERTY, Touch Screen, Mobile.

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 touchscreen 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 [1, 2], this operation is time consuming and error-prone.

To address this problem, in this paper we propose **TypeInBraille**, a novel typing application 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.

2. **THE TYPING TECHNIQUE**

Before presenting the typing technique adopted in **TypeInBraille**, we first briefly introduce the Braille code. 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. 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 1(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 1(b)) reports an upper-case “m” are represented by means of the capital letter symbol, followed by the letter itself.

![Figure 1: Examples of Braille encoding (the black circles stand for raised dots, while the white circles stand for a flat ones).](image)

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, 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 2(a)). Similarly, a tap on the right part corresponds to the right dot raised and the left dot flat (Figure 2(b)). A tap with two fingers represents two raised dots (Figure 2(c)) while a tap with three fingers stands for two flat dots (Figure 2(d)). After a character is entered, it is read by speech to the user and/or a vibration effect is triggered.

![Figure 2: The four gestures defined to enter a pair of dots.](image)

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 3, 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.

![Figure 3: Letter “a” followed by a blank space.](image)

Since the flick gesture is easy to remember and quick to perform, especially by the blind users [2], 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.

TypeInBraille was designed in order to overcome the main limitation of the on-screen QWERTY i.e., the need to search and confirm a small target key. Consequently, TypeInBraille is intuitively more efficient and less error-prone. Another advantage of our solution is that the audio feedback is not indispensable because each gesture is significantly different from the others and hence users can perform the gestures with high confidence, even without the audio feedback. Consequently, with TypeInBraille, the user can also type in noisy environments or when the audio is not available at all.

3. THE TypeInBraille APPLICATION

The TypeInBraille application presents four working modes, which can be set through the rotor gesture: “typing mode”, “exploration mode”, “selection mode” and “sending mode”. In “typing mode”, the user can type text as explained in Section 2 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). Finally, in “sending mode” the user can choose, from a menu, to copy, cut and paste the text, or to send it as email or text message.

4. THE DEMO PRESENTATION

The demo presentation is divided in two stages. In the first one, the participant is invited to experience the hindrances that a blind user runs into while typing with an on-screen QWERTY keyboard. To achieve this, the participant is asked to type using an iPhone device with Voice Over and with the display turned off. If necessary, we briefly train the participant to type using Voice Over. Typing is performed both with audio feedback and without, hence showing that in absence of the audio feedback the number of errors is so high that the result is not understandable.

In the second stage, we briefly introduce the Braille code as well as the typing technique we propose. Then, we ask the participant to type using TypeInBraille. Again, the display is turned off. In case the participant is not proficient in Braille we support him/her by showing the Braille characters map. During this stage, the participant first experiences the use of TypeInBraille supported by the audio feedback and can appreciate that TypeInBraille is much easier to learn and use than the on-screen QWERTY even for users that are novice to the Braille code. Then, the participant can also try to type with TypeInBraille while the audio is turned off and can then realize that, although the audio feedback is helpful, it is not indispensable. At the end of the demo, the participant can try to send the typed text via email.

5. REFERENCES
