Journal of Interaction Science
a SpringerOpen Journal

RESEARCH Open Access

Efficient computer operation for users with a neuromuscular disease with OnScreenDualScribe

Torsten Felzer[1*], Ian Scott MacKenzie[2] and Stephan Rinderknecht[1]

Abstract

We developed a tool based on a modified number pad to empower persons with certain diseases, in particular of neuromuscular origin, to efficiently operate a computer and enter text. As the keypad lies securely in both hands, the system is ideal for someone who has motor problems using a full-size keyboard. The software offers various assistive techniques. For example, text entry is facilitated with the help of word prediction, and an ambiguous mode with word-level disambiguation allows text entry using the entire Latin alphabet with six keys.

In addition to describing the system, we analyze the ambiguous mode and the influence of dictionary size. Initial empirical results with the system, which is already in operation, indicate that it indeed represents a viable alternative by decreasing effort without increasing the time to operate a computer.

This journal article mainly differs from the related proceedings paper through an extended literature review and analyses regarding dictionary size.

Keywords: Human-computer interaction; Keyboard replacement; Mouse emulator; Word prediction; Ambiguous keyboards; Dysarthria; Neuromuscular diseases; Friedreich's Ataxia

Introduction

Whether for work or leisure, in this day and age, it seems nearly impossible to avoid computers. The activity of entering text—perhaps to write a scientific article, to compose an e-mail, or even to control a video game—is a particularly common interaction in this context. The standard input tools for human-computer interaction consist of a full-size keyboard and mouse. These allow for fast and efficient computer operation, provided that the user is *physically* able to operate them.

Persons with motor-related disabilities are often unable to use a standard keyboard. And so, alternatives have been developed. However, for an individual with motor and speech impairments these solutions are problematic because they lack efficiency. For example, scanning solutions are typically very slow although they require only a single input in the form of a switch activation [1,2]. Another limited option is speech recognition. Text entry using speech can be fast, but speech recognition requires the ability to enunciate clearly. Such a model is not an option for a user with a neuromuscular disease because of dysarthria[a]. Questions therefore arise on how to support users with motor and speech impairments who are forced to invest considerable effort and time to work with a standard keyboard because other options are unacceptably slow or inaccessible? These questions motivated the research presented herein.

With these interaction problems in mind, we developed several alternative systems. Work began nine years ago with a simple onscreen keyboard. The user was expected to move the mouse pointer over large rectangular areas on the screen and to select characters by generating clicks using intentional muscle contractions (IMC's) [3,4]. Text entry was accelerated with word completion, but this did not alleviate the temporal overhead induced by the single-signal input method [5]. Later, we used IMC's in combination with a Morse code-like system [6] and an implementation of a scanning ambiguous keyboard [7]. Each system was faster than the preceding one, but was slow for situations requiring manual input. Finally, we

*Correspondence: felzer@ims.tu-darmstadt.de
[1] Institute for Mechatronic Systems, Technische Universität Darmstadt, Otto-Berndt-Str. 2, 64287 Darmstadt, Germany
Full list of author information is available at the end of the article

arrived at the predecessor of the tool reviewed in this article which was limited to a dedicated editor window and thus not yet a general replacement [8].

We focused on the particular case of a Friedreich's Ataxia (FA) patient (let's call him John), but our goal was to help not just one person, but anyone with similar interaction requirements. FA is an inherited, progressive neuromuscular disease that affects the neural pathways between the brain, cerebellum, and muscles [9]. It leads to impaired muscle coordination. There is currently no cure [10], but the symptoms can be treated. Our newest solution, *OnScreenDualScribe*, is such a treatment since it is intended to reduce the effort required to interact with a computer. The primary motivation was to tailor the system to an input device with a form factor ideal for someone with a neuromuscular disease. The outcome is the numeric keypad shown in Figure 1, which lies securely in both hands. The novelty of the approach was to convert that device into a comprehensive assistive input tool capable of conveniently replacing both the keyboard and mouse.

The remainder of this article is organized as follows. In the next section, we review keyboard replacements methods (primarily for text entry) and pointing alternatives reported in the literature. This is followed by three sections describing our solution. Described are the input device, the software implementation, and the dictionary used in *ambiguous mode*. An initial pilot study is then presented, followed by a discussion of the outcome and a summary of the main points.

Related work

Progress in computer technology also includes the development of new methods to help individuals with disabilities enter text through a keyboard alternative. Earlier systems [11] use basic techniques such as word prediction [12] to facilitate input. Although operating a word prediction utility in the simplest form still relies on keyboard input, such an alternative is, in some sense, just an enhancement of the standard device. Growing computational power makes it possible to analyze spoken words and even video recordings online. So speech recognition [13,14] or eye tracking [15-18] have emerged as alternatives. Eye tracking for text entry amounts to moving a mouse pointer across an on-screen keyboard. Thus, eye tracking is simply a pointing device alternative.

To date, there are methods that use an input device different from the standard keyboard without requiring extensive computational power. Examples include: *EdgeWrite* [19,20], a text entry system based on two-dimensional traces representing individual characters; *Dasher* [21], which also uses a pointing device for text entry; and numerous approaches with small keyboards, such as *MessageEase* [22], an ambiguous keyboard with 12 keys similar to a phone keypad; *Sibylle* [23] and *HandiGlyph* [24], which are switch-activated scanning systems; or *OneKey* and *Qanti* (both [25]) which combine scanning with an ambiguous keyboard.

Unfortunately, for an FA patient, these approaches demand either too much or too little. More precisely, for an individual with FA or a person with fine motor and speech impairments, these devices are based on a generalized disability interaction paradigm. The designs do not address specific needs and capabilities: they either require unimpaired motor (or vocal) skills, or they do not take full advantage of the true abilities of the target population, thus making computing tasks unnecessarily slow.

Returning to our FA user, John, he has problems using a full-size keyboard, but cannot use most hands-free alternatives either. The most important example is speech recognition. Due to dysarthria, the variability in his speech is too large for recognition software. The problem for an algorithm to learn John's phonemic patterns is exacerbated by the difficulty he experiences to produce humming sounds. The generation of such para-language is required for using NVVI, (Non-Verbal Vocal Interaction)

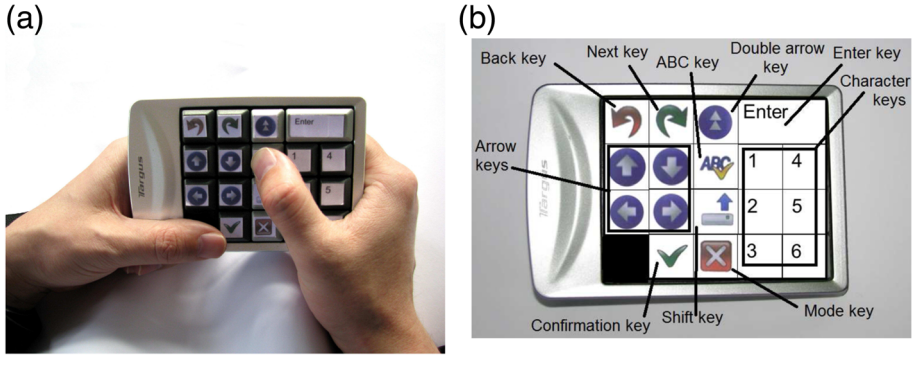

Figure 1 *DualPad* **2.0 inexpensive 18-key wireless number pad: (a) usage: user can reach every key with the thumbs without repositioning the hands; left and right edges serve as tactile guide when aiming at the keys; (b) stickers for the 18 keys with explanations.**

[26-29]. A secondary symptom that John experiences is a pathological nystagmus, which causes his eye movements to be rather unusual. In contrast to physiological nystagmus, pathological nystagmus causes drift of retinal images and degraded vision for lack of compensatory eye movements [30]. In summary, John experiences involuntary eye movements and reduced ocular muscle control, which is a combination of symptoms that obviously renders eye tracking approaches unusable.

On the other hand, John's capabilities go beyond single-switch input. Yet, there are limitations to his ability to use a full-size keyboard. When trying to navigate from key to key in a "free-hand" manner, he must frequently reposition his hands and aim at each key. In doing so, he often hits the wrong key or the right key in an imprecise fashion causing extra letters to inadvertently appear. However, he is indeed capable of using both hands. When properly guided, he can grasp the left and right edge of a regular keyboard and enter text with difficulty but with fewer mistakes. Clearly a keyboard replacement that does not leverage his capabilities is not a true alternative, since the resulting interaction is unnecessarily slow. There exist some text entry alternatives for persons with disabilities that are usable by John. An example is the limited-vocabulary speech recognition alternative of Hamidi et al. [31], but such approaches generally do not support editing text and rearranging sentences. With limited functionality, such methods can hardly be called replacements.

The situation for pointing device alternatives is similar: At one end of the spectrum there are utilities allowing control of the mouse pointer with the help of a standard keyboard [32]. At the other end are head tracking systems [33-35], speech recognition [36], or NVVI [37]. In addition, there are many hardware solutions to control the mouse pointer. Besides the standard mouse, there are trackballs, touchpads, or joysticks. As already noted, due to loss of fine motor coordination, someone with a neuromuscular disease cannot use systems based on motion tracking. Moreover, manual pointing devices are sometimes problematic, since the user often has to switch between text entry and pointing operations. In order to be able to do that quickly and efficiently, users of manual devices usually need one hand free.

Following this overview it is apparent that an efficient solution for John (and individuals with similar needs) involves combining keyboard and mouse replacements in a single manual device. Our system presented here and previously [38] is intended as a computer interaction device for patients with FA or individuals with similar symptomatologies or interactions needs.

Hardware specifics

We wanted the input device for *OnScreenDualScribe* to be inexpensive, readily available, and convenient to handle. During research and development, it became clear that a good choice is a small, off-the-shelf number pad operated sideways and modified with keytop stickers for the remapping. After testing seven to eight different number pads, we selected the wireless device in Figure 1a. The wireless option was chosen to maximize user convenience. The stickers and the corresponding key layout are shown in Figure 1b.

The left two key columns are operated with the left thumb and the remaining keys with the right thumb. The depicted layout is used in most contexts with the *ambiguous mode* as the only exception. In *ambiguous mode* the key assignments are slightly different. This is necessary because all character keys are in columns 4 and 5. Without changes, the right thumb would do all the typing while the left thumb would be idle. The actually applied stickers were modified from Figure 1b to reflect both layouts[b].

Software overview

OnScreenDualScribe is written in C++ under the Windows® operating system. It captures physical keystrokes on the number pad but does not pass them through to the active window. This is accomplished by defining all keys as "Hotkeys". Instead, the software sends virtual input events to the active application. More details are given below.

General architecture

The software operates in nine modes, each of which is responsible for different computing tasks. The 18 keys of the *DualPad* are remapped, depending on the currently active mode. The program window is shown as a narrow vertical stripe on top of all other windows; that is, it is always visible at the left or right edge of the screen (see Figure 2a). It displays several mode-specific indicators, for instance, which key is mapped to which function in the current mode.

To help novice users with the key associations, the key indicators resemble the physical layout of Figure 1b. Two examples are illustrated in Figure 3. These show the indicators for *function mode* and *mouse mode* with similar left-hand keys.

Some of the *DualPad* keys directly lead to the generation of virtual keystrokes or mouse events[c]. For example, pressing the "Confirmation key" on the *DualPad* in most modes generates an input event corresponding to the RETURN key on a standard keyboard. Other keys either change the program state, for example, in the course of selecting a word candidate (see below) or alter the currently active mode.

Text entry

There are two methods for entering text in *OnScreenDualScribe*. The basic method, called *dual mode*, allows users to "write" the characters presented in the two-dimensional

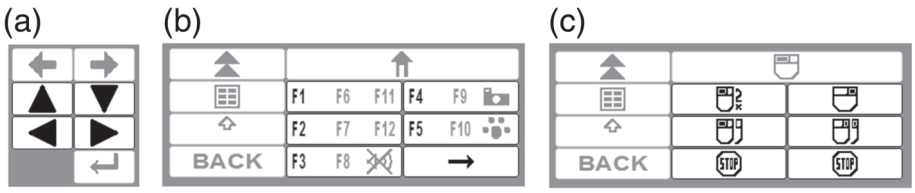

Figure 2 Indicators for the keys in function mode and mouse mode: (a) left-hand keys for scrolling (function mode) or moving the pointer (mouse mode) and navigating in a web browser; (b) right-hand keys in function mode for evoking special virtual keystrokes; (c) right keys in mouse mode for generating clicks (left, right, single, double, drag).

8×6 virtual keyboard of Figure 4a. This is accomplished by selecting a row (with one or two keystrokes on the four arrow keys) and another keystroke on the character key that corresponds to its column.

Despite the labeling, the arrow keys are not used to move the highlight marking the selected row up or down. Rather, striking arrow key a_i once, directly highlights row i, and striking it twice highlights row $i + 4$ ($i = 1, \ldots, 4$). This type of selection scheme is demonstrated in Figure 4b. Figure 4c shows the selected row on the character keys.

With the exception of "y" and "z", all letter characters require two keystrokes. Since the highlighted row resets to row five after choosing a character, the *KSPC* (keystrokes per character) for this input method is slightly less than 2.0. However, after each character, *OnScreenDualScribe* looks up the entered word stem in a dictionary and presents a frequency-ordered list of extensions. The dictionary is a text file containing 100,000 words for the English version. The words are from the Corpus of Contemporary American English [39]. Keystrokes are saved if the intended word is in the list. Selection of a suggested word is done analogous to the selection of a row of the virtual keyboard: The word list is divided into two sublists, and like previously described interactions, one of the eight rows/entries is highlighted in either sublist (see Figure 4d, Figure 5c, and Figure 5d). Final selection involves choosing between the two highlighted words by striking one of two particular keys.

To some extent, word prediction brings this "open" text entry technique close to a regular QWERTY keyboard with a *KSPC* of 1.0. Applying

$$KSPC = \frac{\sum_{i=1}^{n} \left(K_{w_i} \cdot F_{w_i}\right)}{\sum_{i=1}^{n} \left(L_{w_i} \cdot F_{w_i}\right)}, \tag{1}$$

to the newest edition of the dictionary file, the resulting *KSPC* is 1.1169 for *dual mode*. Note: K_{w_i} is the least number of keystrokes required to enter the word w_i, F_{w_i} is the frequency of w_i in the corpus, and L_{w_i} is the number of characters of w_i [40].

The second text entry method uses an ambiguous keyboard with six character keys and dictionary-based disambiguation. This is called *ambiguous mode*. In this mode, character keys are used to compose a sequence of "code characters" represented by the digits 1 to 6. Each code character represents four or five characters of the alphabet (see Figure 2b). As the user types code characters, the software matches the code sequence among the words in the same dictionary file as above. The resulting frequency-ordered list of matching candidates contains at most 16

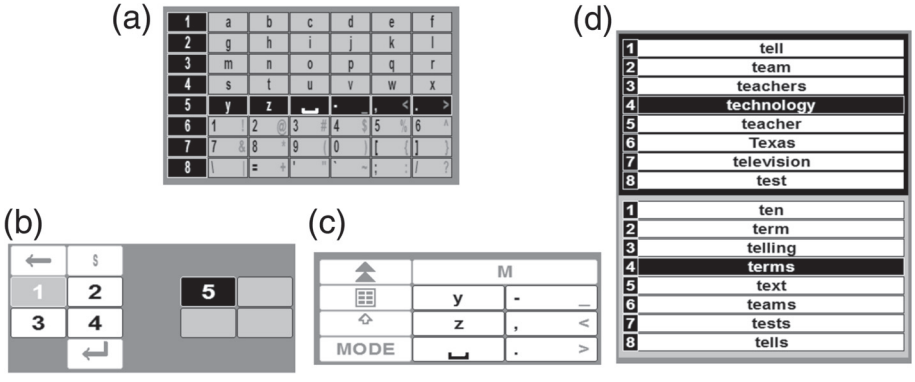

Figure 3 Text entry in dual mode: (a) virtual keyboard with eight rows and six columns showing the available characters (shifted versions for non-letter characters are shown in gray); (b) left-hand keys – arrow keys (here labeled "1" – "4", cf. Figure 2a) are used to highlight one of the eight rows of the virtual keyboard; **(c)** right-hand keys – character keys correspond to the highlighted row; **(d)** word prediction list after entering "te"İ (i.e., "!2"5").

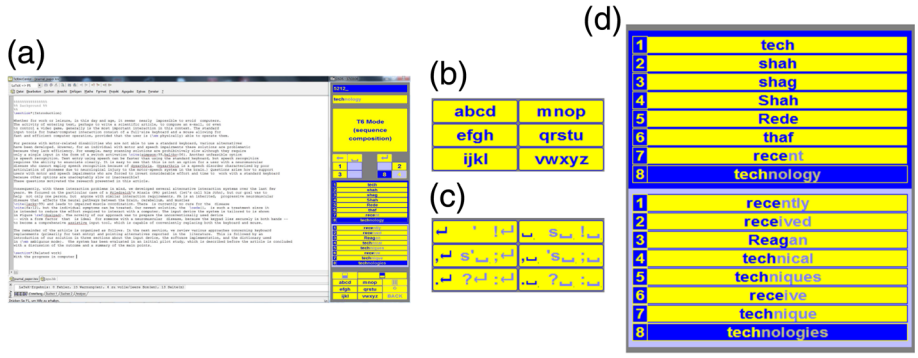

Figure 4 Ambiguous mode: (a) desktop snapshot with program window at the right; (b) default arrangement where 26 letters are distributed over six keys in alphabetic order; (c) options following a selected candidate; (d) candidate list for the sequence "5212".

entries (selection as before), including extensions if there are less than 16 candidates equal in length to the code sequence (see Figure 2d).

This mode is more efficient than *dual mode* (*KSPC* is only 0.8678), but entering out-of-dictionary words is not possible. After selecting a candidate, the user chooses among several options for finalizing the selection (see Figure 2c) by adding a space or basic punctuation. This reduces the need to switch to the basic technique between words.

Performing pointing device operations

In addition to the use of a keyboard, certain computing tasks, such as switching applications or clicking on a URL, benefit from the use of a pointing device. Since the *DualPad* is held with both hands, using an additional device for pointing (like a mouse or a trackball) requires repositioning the hands. This approach is inefficient due to the increase in the user's motor workload.

Therefore, it was considered necessary to include a *mouse mode* (see Figure 3c). The challenge was to find a solution that allowed precise pointer control, while permitting fast movement across the screen at the same time. To do this, *OnScreenDualScribe* continually moves the pointer in one direction after the corresponding arrow key is pressed. The initial speed is low (ten pixels per second) and is reset when a different arrow key is pressed. However, pressing the same arrow key again accelerates the movement. When the pointer reaches the desired destination, the user generates a click or stops the movement with the right-hand keys.

Additional functionality

Operating a computer requires keyboard functionality that is beyond plain text entry. *OnScreenDualScribe* provides several ways to use special keys, like SHIFT, CTRL, or ALT. An example is *single keystroke mode*. In this mode, a single keystroke directly triggers a virtual keystroke (as

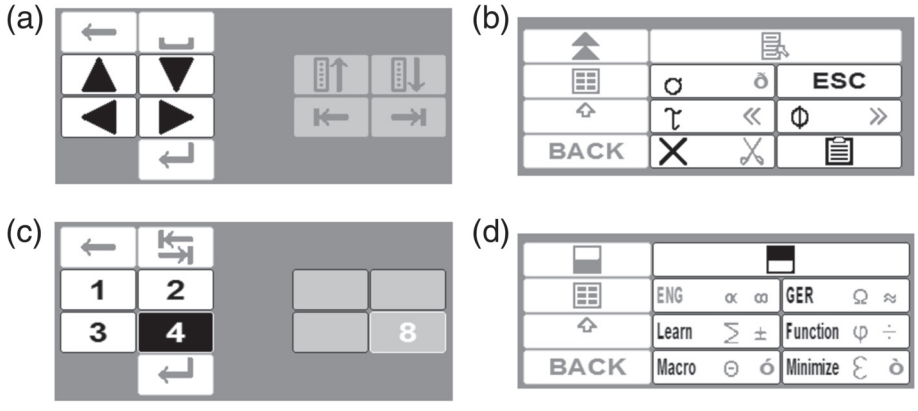

Figure 5 Key indicators in *single keystroke mode*: (a) arrow keys to relocate text cursor; (b) right-hand keys for common keyboard functions; and *menu mode*: (c) arrow keys to control the highlights in word prediction sublists; (d) right-hand keys to select desired sublist or trigger other modes.

opposed to *dual mode*). It gives access to frequently used keys, like the regular arrow keys, ESCAPE, DELETE, or PASTE (Figures 5a and 5b).

Finally, there is a *macro mode* for sending arbitrary keystroke sequences, a *spell check mode* to check the spelling of an entered word, and a *learning mode* to build and update a personal user-dependent dictionary supplement. These modes are activated by the *menu mode* which also allows users to select word prediction candidates or to toggle between English and German at runtime (Figures 5c and 5d). The German dictionary is based on a source from the Institute for the German Language [41]. However, all statistical data given in our article (such as *KSPC*) refer to the English version only.

A demo of the software is presented in a video accompanying the article (see Additional file 1).

Development and considerations regarding the dictionary

The purpose of an ambiguous keyboard is to provide access to all characters in a given alphabet A with less than $|A|$ keys. In *OnScreenDualScribe*, $A = \{c_1, c_z, \ldots, c_{26}\}$ is the set of Latin characters and is spread over six keys. As 6 is considerably less than 26, this is not a trivial task. One way to disambiguate sequences of keys is to look for matches in a dictionary $D = \{w_1, w_2, \ldots, w_n\}$ with n words (for *OnScreenDualScribe*, n = 100,000).

Particularly for an assistive tool, it is important to minimize the number of times the user starts entering a sequence and later realizes that the intended word is not in the dictionary, necessitating re-entry[d].

In other words, the dictionary should be large. But there is a tradeoff: With a set number of ambiguous keys, a larger dictionary yields more matches and thus longer candidate lists. As a consequence, as *OnScreenDualScribe* at most presents the top 16 candidates, words with a frequency ranked 17 or more (among all words of the same length) cannot be selected. Therefore, the dictionary size is limited.

To determine if there is a problem with this in the initial configuration of *OnScreenDualScribe*, we computed the rank of the code sequence for each word (after it is fully entered). The result for the default arrangement in the system's *ambiguous mode* is depicted in Figure 6a.

Considering the relative frequencies of words in the corpus underlying the dictionary, the intended word appears at the top of the list with a probability of 89%. In these cases, selection simply requires confirmation. The remaining 11% of cases require active selection. Only with a probability of about 0.03% (1,800 words), words cannot be selected at all.

The assignment of letters to keys for the ambiguous keyboard influences the candidate lists. Lesher et al. describe an algorithm to optimize the arrangement of an ambiguous keyboard [42], but only using character-level disambiguation. Besides, there are

$$\binom{26}{4} \cdot \binom{22}{4} \cdot \binom{18}{4} \cdot \binom{14}{4} \cdot \binom{10}{5} \cdot \binom{5}{5} \cdot \frac{1}{6!} = \frac{26!}{4!^4 \cdot 5!^2 \cdot 6!} \approx 1.17 \times 10^{14}$$

different arrangements with 26 characters distributed over six keys (counting only those with four 4-character keys and two 5-character keys). If a fast computer needed one second to check[e] an arrangement (by generating the equivalent of Figure 6a), an exhaustive search would take more than three million years.

Therefore, what remains are heuristic considerations: Code sequences that are more frequent obviously produce longer candidate lists. Since the frequency of a code sequence is directly linked to the frequency of the keys in that sequence, it follows that frequently used keys (with frequent characters) tend to cause problems. In the default arrangement, the fifth key (representing "r", "s", and "t") is presumably an example of such a frequent key.

To quantify that presumption, we computed the relative frequency F_C of a character C within the underlying corpus using

$$F_C = \frac{\sum_{i=1}^{n} \left(NC_{w_i} \cdot F_{w_i} \right)}{\sum_{i=1}^{n} \left(L_{w_i} \cdot F_{w_i} \right)}, \tag{2}$$

where NC_{w_i} is the number of times the character C occurs in the word w_i, F_{w_i} and L_{w_i} as in equation (1).

The resulting relative frequencies for all 26 characters are displayed in Figure 7. Accumulating those values for characters lying on the same key leads to Figure 8a. It is clearly seen that the fifth key (qrstu) is struck five times as often as the sixth key (vwxyz). This reduces the efficiency of the disambiguation.

With the objective of making each key almost equally probable, we arrived at the arrangement in Figure 8b (which also shows the corresponding key occurrences).

However, recomputing the distribution in Figure 6a gives only a minimum of improvement (see Figure 6b). The share of top candidates is now about 90%, with other selectable candidates occurring in 10% of all cases. Words corresponding to invalid sequences only occur with 0.02% probability. These words are not in the candidate list even though the corresponding code sequence is fully entered. In absolute terms, there are 1,341 such words.

In addition, *KSPC* only reduces by seven thousandths to 0.8605. Since using a non-alphabetic arrangement requires additional training from the user, it seemed reasonable not to further investigate this optimization problem.

One last consideration is that it could be expected from the size of the dictionary that there is little room for

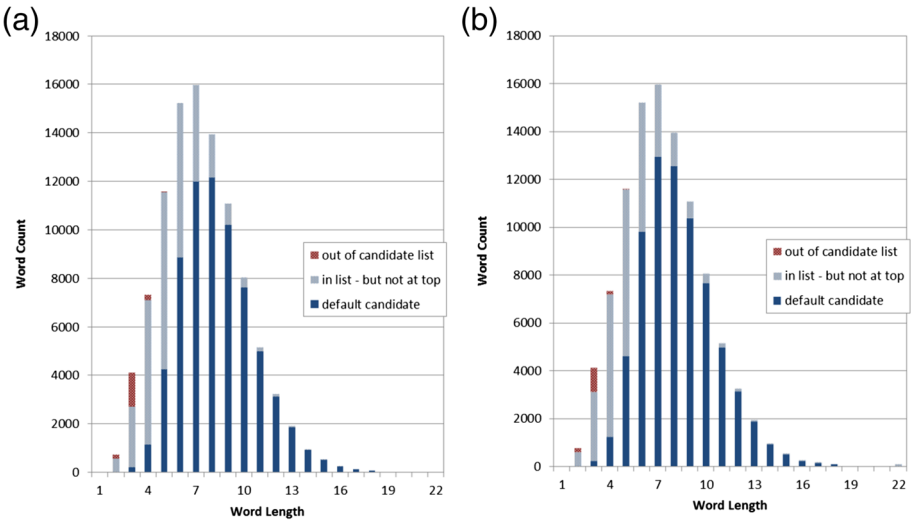

Figure 6 Distribution of 100,000 dictionary words according to length; bars also show how well the software can disambiguate full-length code sequences: (a) for default arrangement; (b) for arrangement in Figure 8b; the difference is minimal (e.g., the top result for 7-character words is achieved in about 12,000 (*left*) vs. 13,000 (*right*) cases).

improvement. For example, independent of the arrangement, *OnScreenDualScribe* gives access to a maximum of $6 \times 6 \times 6 \times 16 = 3{,}456$ 3-letter candidates. As the dictionary contains more than 4,000 3-letter words, some of them *must* be invalid. However, as the dictionary is also used for word prediction and spell-checking, it makes sense not to delete these words.

Methods

The system was originally built for John, a 41-year-old FA patient, who regularly uses *DualScribe* [8], a predecessor system which is limited to a proprietary editor window and is thus unable to control arbitrary applications. John helped evaluate a simpler variant of the tool, which was based on a game controller [43]. A pilot study involved entering portions of a 2,100-character text over the course of five days for at least two hours per day.

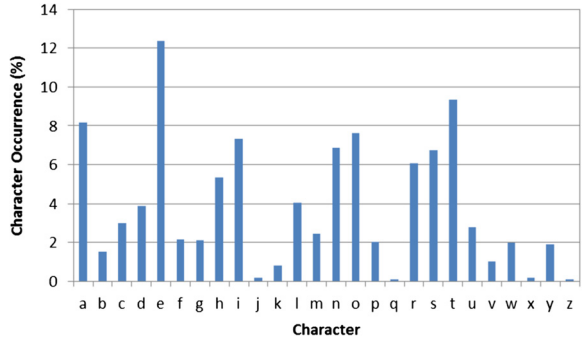

Figure 7 Relative frequencies of all 26 characters in the test dictionary (see text for explanation).

He repeated the experiment with the new system, only this time entering the entire text each day (which almost always took less than 120 minutes). On the old system, his entry rate was approximately 2 wpm, and with the new system, his performance improved to over 3.5 wpm. With the standard keyboard, John is able to achieve entry rates between 2 wpm and 4 wpm, but this demands more effort than *OnScreenDualScribe* due to frequent typing errors and the need to operate a full-size and cumbersome keyboard.

Furthermore, the first author included a very unconventional form of empirical evaluation: About 90% of this article was "written" with *OnScreenDualScribe*. The fact that the first author did not switch to the standard keyboard after one or two days certainly speaks to the quality of the interface.

Results and discussion

OnScreenDualScribe allows the user to enter text with fewer keys and mostly with fewer keystrokes than usually needed, thereby reducing the user's physical workload (also [44]). Some aspects, such as knowing which key to press next, when to look at the candidate lists, or deciding if and where the intended word is present may somewhat increase cognitive load although mnemonic support is provided.

In addition, the letter layout of the 8 × 6 grid *and* that of the character keys in *ambiguous mode* is relatively easy to remember, since they are approximately alphabetic. As far as *ambiguous mode* is concerned, this choice has been justified in the section on the dictionary size: a frequency-based arrangement results in marginal

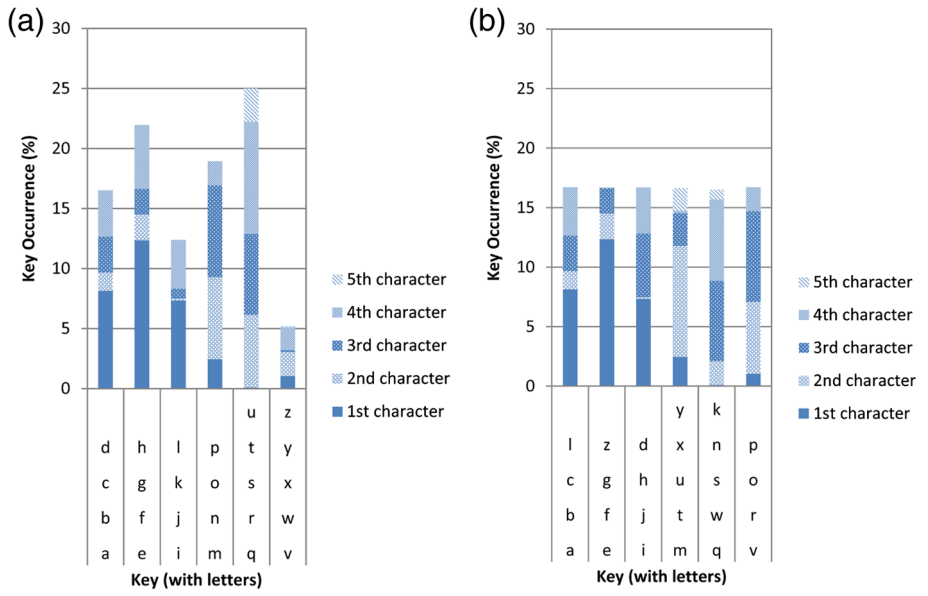

Figure 8 Cumulative occurrence of keys (a) default arrangement; (b) alternative arrangement, based on character frequencies—each key's occurrence is between 16.51% and 16.73%.

improvements regarding disambiguation quality, but this does not outweigh the fact that an alphabetic arrangement is much more intuitive. However, remembering the letter to key assignments still requires some practice.

In any event, keeping in mind that *OnScreenDualScribe* was designed for users with fine motor and speech challenges (and non-impaired cognitive functioning), the test of the usefulness of the device rests with the user, for example, John.

In fact, John switched to the tool and its predecessor as soon as they became available. His initial interactions were limited to composing emails, then expanded quickly to encompass all his computer interaction needs. By now, the tool is John's indispensable assistant: His life is easier! He notes, "Of course, I have to concentrate, and when I am tired, my writing speed drops, but taking away the effort is much more important for me". His self-reports are supported by quantitative measures: John's entry rate (more than half a year after the above mentioned experiment) improved to typically between 3 wpm and 5 wpm. Considering that reduced vigilance is coincident with lower entry rates while using the standard keyboard, John is faster with the alternative method at any given point in time.

John has become so proficient in using the program that he can anticipate the number of letters required before a word appears near the top of the candidate list in *ambiguous mode*. He also developed strategies to easily circumvent problems like out-of-dictionary words. For example, the word "neuromuscular" is not in the dictionary, but the words "neuro" and "muscular" are; John knows this and he can quickly start a new word after having entered "neuro" when trying to enter the compound word in *ambiguous mode*. This is considerably faster than entering the entire sequence, just to see that the word is *still* not in the list.

The quantitative and anecdotal evidence of the learning curve in the interactions illustrates why conducting a usability study with more participants is particularly challenging. Non-expert users must practice with the software for a longer time (on the order of months) to be able to make full use of the tool's assistive power. Finding participants willing to take on such a long-term commitment and controlling attrition is not a trivial task. However, the practical significance and value of the assistive device, even if limited to a few individuals, makes *OnScreenDualScribe* a viable interaction tool that enhances the quality of work, leisure and life of its user.

Conclusion

This article reviewed an assistive computer application and device, called *OnScreenDualScribe*, which replaces a full-size keyboard and a mouse with an inexpensive, commercially available number pad. The system has been developed with the initial goal to provide a practical and efficient solution for John, an individual with FA who participated in earlier studies evaluating text entry alternatives. John also took part in a pilot study testing *OnScreenDualScribe*.

The broader idea is to help anyone with similar conditions and interaction needs. Since *OnScreenDualScribe* is based on a small, compact manual device, it represents a viable alternative for persons with neuromuscular diseases. Such individuals are often unable to use speech

recognition, but are able (within limits) to use both hands. Furthermore, able-bodied users looking for a small-size keyboard (e.g., for browsing the Internet on a television) may also be interested in using the system. The interactions supported by *OnScreenDualScribe* integrate the functionality of a keyboard and pointing device in a single device. Instrumental to the research and development was to work closely with end-users to identify and meet their requirements—requirements not addressed by existing disability interaction models.

After the pilot study, John immediately decided to switch all his interactions to the *OnScreenDualScribe*. This transition included the use of an eye-catching novelty in his workplace—a new 33" monitor. John notes that it is "not a necessity, but a nice addition to my computer equipment, making it easier to scan the word lists". A standard keyboard and mouse are still on the table, "but only as a fallback option, in case Windows hangs".

The most important task for the future involves conducting a larger usability study. The next steps also include debugging and improving the software and designing a proprietary input device that supports universal access.

Consent

Written informed consent was obtained from the patient for the publication of this report and any accompanying images.

Endnotes

[a]Dysarthria is a speech disorder characterized by poor articulation of phonemes due to neurological injury to the motor-speech system in the brain [45].

[b]As described below, users are not expected to remember key mappings but a mnemonic aide is provided during interaction.

[c]See the subsection on pointing device operations.

[d]In *OnScreenDualScribe*, this would probably involve using *dual mode*.

[e]In fact, the computer we used—an Intel® i3 at 2.13 GHz with 4 GB RAM—was busy for more than ten minutes.

Additional file

Additional file 1: To demo *OnScreenDualScribe*, several sentences are transcribed. Both *dual mode* (plus word prediction) and *ambiguous mode* are used two times each: Slow at first to show how the tool works, and then fast to demo what it can do.

Competing interests

The authors declare that they have no competing interests.

Author's contributions

The first author programmed *OnScreenDualScribe*, produced the demo video, and wrote a preliminary version of the paper. The second author helped proofreading, editing, improving, and extending the manuscript. *OnScreenDualScribe*'s *ambiguous mode* was originally inspired by the second author's *OneKey* application [25]. All authors read and approved the final manuscript.

Acknowledgements

This work is partially supported by DFG grant FE 936/6-1 "EFFENDI—EFficient and Fast text ENtry for persons with motor Disabilities of neuromuscular orIgin". Parts of this article appeared as a conference paper in the *Proceedings of HCII 2013* (Felzer, T., MacKenzie, I.S., Rinderknecht, S.: OnScreenDualScribe: A computer operation tool for users with a neuromuscular disease. In: Proc. HCI International 2013, UAHCI/HCII 2013, Part I, LNCS 8009, pp. 474–483). Springer, Heidelberg (2013).

Author details

[1]Institute for Mechatronic Systems, Technische Universität Darmstadt, Otto-Berndt-Str. 2, 64287 Darmstadt, Germany. [2]Department of Electrical Engineering and Computer Science, York University, M3J 1P3 Toronto, Canada.

Received: 8 May 2014 Accepted: 9 May 2014
Published online: 25 June 2014

References

1. Simpson, RC, & Koester, HH (1999). Adaptive one-switch row-column scanning. *Rehabilitation Engineering, IEEE Transactions on*, *7*(4), 464–473.
2. Baljko, M, & Tam, A (2006). Motor input assistance: indirect text entry using one or two keys, In *Proc. ASSETS 2006* (pp. 18–25). New York: ACM.
3. Felzer, T, Fischer, R, Grönsfelder, T, Nordmann, R (2005). Alternative control system for operating a PC using intentional muscle contractions only, In *Online-Proc. CSUN Conf. 2005*. Northridge, CA, USA: California State University.
4. Felzer, T, & Nordmann, R (2008). Evaluating the hands-free mouse control system: sn initial case study, In *Proc. ICCHP 2008, vol. 5105/2008* (pp. 1188–1195). Heidelberg: Springer.
5. Felzer, T, & Nordmann, R (2006). Speeding up hands-free text entry, In *Proc. CWUAAT 2006* (pp. 27–36). Cambridge: Cambridge University Press.
6. Felzer, T, & Nordmann, R (2006). Alternative text entry using different input methods, In *Proc. ASSETS 2006*. New York: ACM.
7. Felzer, T, MacKenzie, IS, Beckerle, P, Rinderknecht, S (2010). Qanti: a software tool for quick ambiguous non-standard text input, In *Proc. ICCHP 2010* (pp. 128–135). Heidelberg: Springer.
8. Felzer, T, MacKenzie, IS, Rinderknecht, S (2012). DualScribe: a keyboard replacement for those with Friedreich's Ataxia and related diseases, In *Proc. ICCHP 2012* (pp. 431–438). Heidelberg: Springer.
9. Lecky, BRF (1999). Neuromuscular disorders: clinical and molecular genetics. *Brain*, *122*(4), 1–790.
10. National Institute of Neurological Disorders and Stroke. Friedreich's Ataxia Fact Sheet. http://www.ninds.nih.gov/disorders/friedreichs_ataxia/detail_friedreichs_ataxia.htm. Accessed 27 April 2012.
11. Higginbotham, DJ (1992). Evaluation of keystroke savings across five assistive communication technologies. *Augmentative & Alternative Communication*, *8*, 258–272.
12. Koester, HH, & Levine, SP (1994). Modeling the speed of text entry with a word prediction interface. *Rehabilitation Engineering, IEEE Transactions on*, *2*(3), 177–187.
13. Zhang, W, Duffy, VG, Linn, R, Luximon, A (1999). Voice recognition based human-computer interface design. *Computers & Industrial Engineering*, *37*, 305–308.
14. Hirsimäki, T, & Kurimo, M (2009). Analysing recognition errors in unlimited-vocabulary speech recognition, In *Proc. NAACL 2009* (pp. 193–196). Stroudsburg: Association for Computational Linguistics.
15. Kristensson, PO, & Vertanen, K (2012). The potential of dwell-free eye-typing for fast assistive gaze communication, In *Proc. ETRA 2012* (pp. 241–244). New York: ACM.
16. Ashtiani, B, & MacKenzie, IS (2010). BlinkWrite2: an improved text entry method using eye blinks, In *Proc. ETRA 2010* (pp. 339–345). New York: ACM.
17. Zhao, XA, Guestrin, ED, Sayenko, D, Simpson, T, Gauthier, M, Popovic, MR (2012). Typing with eye-gaze and tooth-clicks, In *Proc. ETRA 2012* (pp. 341–344). New York: ACM.

18. Špakov, O, & Miniotas, D (2004). On-line adjustment of dwell time for target selection by gaze, In *Proc. NordiCHI 2004* (pp. 203–206). New York: ACM.
19. Wobbrock, JO, Myers, BA, Kembel, JA (2003). EdgeWrite: a stylus-based text entry method designed for high accuracy and stability of motion, In *Proc. UIST 2003* (pp. 61–70). New York: ACM.
20. Wobbrock, JO, Myers, BA, Aung, HH, LoPresti, EF (2004). Text entry from power wheelchairs: EdgeWrite for joysticks and touchpads, In *Proc. ASSETS 2004* (pp. 110–117). New York: ACM.
21. Ward, DJ, Blackwell, AF, MacKay, DJC (2000). Dasher - a data entry interface using continuous gestures and language models, In *Proc. UIST 2000* (pp. 129–137). New York: ACM.
22. Nesbat, SB (2003). A system for fast, full-text entry for small electronic devices, In *Proc. ICMI 2003* (pp. 4–11). New York: ACM.
23. Wandmacher, T, Antoine, J-Y, Poirier, F, Départe, J-P (2008). Sibylle, an assistive communication system adapting to the context and its user. *ACM Transactions on Accessible Computin*, *1*(1), 1–30.
24. Belatar, M, & Poirier, F (2008). Text entry for mobile devices and users with severe motor impairments: HandiGlyph, a primitive shapes based onscreen keyboard, In *Proc. ASSETS 2008* (pp. 209–216). New York: ACM.
25. MacKenzie, IS, & Felzer, T (2010). SAK: Scanning Ambiguous Keyboard for efficient one-key text entry. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *17*(3), 11:1–11:39.
26. Watts, R, & Robinson, P (1999). Controlling computers by whistling, In *Proceedings of Eurographics UK*. Cambridge, United Kingdom: Cambridge University Press.
27. Sporka, AJ, Kurniawan, SH, Slavík, P (2004). Whistling user interface (u3i), In *8th ERCIM International Workshop "User Interfaces For All", LCNS 3196, Vienna* (pp. 472–478). Heidelberg: Springer.
28. Poláček, O, Míkovec, Z, Sporka, AJ, Slavík, P (2010). New way of vocal interface design: Formal description of non-verbal vocal gestures, In *Proc. CWUAAT 2010* (pp. 137–144). Cambridge: Cambridge University Press.
29. Sporka, AJ, Felzer, T, Kurniawan, SH, Poláček, O, Haiduk, P, MacKenzie, IS (2011). Chanti: predictive text entry using non-verbal vocal input, In *Proc. CHI 2011* (pp. 2463–2472). New York: ACM.
30. Leigh, RJ, & Rucker, JC (2005). Nystagmus and related ocular motility disorders, In *Walsh and Hoyt's Clinical Neuro-Ophthalmology*. 6th edn. Section III, The Ocular Motor System, vol. 1 (pp. 1133–1173). Philadelphia: Lippincott Williams & Wilkins. Chap. 23.
31. Hamidi, F, Baljko, M, Livingston, N, Spalteholz, L (2010). CanSpeak: a customizable speech interface for people with dysarthric speech, In *Proc. ICCHP 2010* (pp. 605–612). Heidelberg: Springer.
32. RH Designs. Mouse Emulator. http://rhdesigns.browseto.org/mouseemulator.html. Accessed 30 April 2012.
33. Gips, J, Betke, M, Fleming, P (2000). The camera mouse: preliminary investigation of automated visual tracking for computer access, In *Proc. RESNA 2000* (pp. 98–100). Arlington: RESNA Press.
34. Kumar, M, Paepcke, A, Winograd, T (2007). EyePoint: practical pointing and selection using gaze and keyboard, In *Proc. CHI 2007* (pp. 421–430). New York: ACM.
35. Javanovic, R, & MacKenzie, IS (2010). MarkerMouse: mouse cursor control using a head-mounted marker, In *Proc. ICCHP 2010* (pp. 49–56). Heidelberg: Springer.
36. Loewenich, F, & Maire, F (2007). Hands-free mouse-pointer manipulation using motion-tracking and speech recognition, In *Proc. OZCHI 2007* (pp. 295–302). New York: ACM.
37. Bilmes, JA, Li, X, Malkin, J, Kilanski, K, Wright, R, Kirchhoff, K, Subramanya, A, Harada, S, Landay, JA, Dowden, P, Chizeck, H (2005). The vocal joystick: a voice-based human-computer interface for individuals with motor impairments. Technical Report UWEETR-2005-0007, University of Washington. https://www.ee.washington.edu/techsite/papers/refer/UWEETR-2005-0007.html.
38. Felzer, T, MacKenzie, IS, Rinderknecht, S (2013). OnScreenDualScribe: a computer operation tool for users with a neuromuscular disease, In *Proc. HCI International 2013, UAHCI/HCII 2013, Part I, LNCS 8009* (pp. 474–483). Heidelberg: Springer.
39. Davies, M. Word frequency data from the Corpus of Contemporary American English (COCA). Downloaded from http://www.wordfrequency.info. January 27 2011.
40. MacKenzie, IS (2002). KSPC (keystrokes per character) as a characteristic of text entry techniques, In *Proc. Mobile HCI 2002* (pp. 195–210). Heidelberg: Springer.
41. Institut für Deutsche Sprache (2009). *Korpusbasierte Wortformenliste DeReWo, V-100000t-2009-04-30-0.1, Mit Benutzerdokumentation*. Programmbereich Korpuslinguistik: Mannheim. http://www.ids-mannheim.de/kl/derewo/.
42. Lesher, GW, Moulton, BJ, Higginbotham, DJ (1998). Optimal character arrangements for ambiguous keyboards. *Rehabilitation Engineering, IEEE Transactions on*, *6*(4), 415–423.
43. Felzer, T, & Rinderknecht, S (2011). Using a game controller for text entry to address abilities and disabilities specific to persons with neuromuscular diseases, In *Proc. ASSETS 2011* (pp. 299–300). New York: ACM.
44. MacKenzie, IS (2013). *Human-Computer Interaction: An Empirical Research Perspective*: Elsevier.
45. Duffy, JR (2013). *Motor Speech Disorders: Substrates, Differential Diagnosis, and Management, 3rd edn. St*. Louis: Elsevier Mosby.

doi:10.1186/s40166-014-0002-7
Cite this article as: Felzer *et al.*: **Efficient computer operation for users with a neuromuscular disease with OnScreenDualScribe.** *Journal of Interaction Science* 2014 **2**:2.