Exploring Russian Tap-Code Text Entry Adaptions for Users with Reduced Target Hitting Accuracy

Frode Eika Sandnes
Department of Computer Science, Faculty of Technology, Art and Design, Oslo and Akershus University College of Applied Sciences
P.O. Box 4, St. Olavs plass
0130 Oslo, Norway
+47 67 23 50 03
Frode-Eika.Sandnes@hioa.no

Fausto Orsi Medola
Laboratory of Ergonomics and Interfaces, Department of Design, Faculty of Architecture, Arts and Communication, UNESP
Bauru, São Paulo, Brazil
+55 14 3103-6062
fausto.medola@faac.unesp.br

ABSTRACT
Text is still the dominant form of human-computer-human communication. Users with certain motor or visual impairments may be unable to use certain text entry interfaces such as the small virtual keyboards on mobile phones effectively due to challenges hitting small targets. Despite the vast amount of research into text entry, no efforts have explored the so-called Russian tap-codes, or knock codes, which were commonly used to communicate between prison inmates. Tapping does not require the user to hit a specific target. This study proposes a theoretical framework for classifying text entry designs. The framework is used to explore 16 text entry designs, namely the classic Russian tap code and design variations exploiting more recent results in text entry research and the context of current hardware, allowing unfeasible designs to be easily eliminated.

CCS Concepts
• Human-Centered computing →Accessibility→Accessibility design and evaluation methods • Human computer interaction (HCI)→HCI design and evaluation methods→Heuristic evaluations.

Keywords
Motor impairment; low-vision; target hitting; text entry; accessibility.

1. INTRODUCTION
Although user interfaces increasingly rely on non-textual interaction such as voice [1] and gestures [2], a majority of interaction between man and machine still depends on text entry, especially as computers are used to mediate written communication between humans. Text entry is an active research field that is driven by constant advancements in input device technology, especially as the form factors has changed from the desktop to mobile and smart devices. Advances in text entry technology aimed at the general population has also benefitted users with various disabilities. Moreover, assistive technologies, such as text prediction [3], have also benefited non-disabled users.

The literature on text entry is vast. After the invention of the typewriter, many alternative keyboard layout designs were explored [4]. Moreover, chording was early proposed as a means for more efficient text input, where instead of hitting individual keys to produce the letters, key patterns where pressed to recall letters [5, 6]. Unfortunately, chording did not succeed, probably due to the effort needed to learn chords and a lack of a standard. One exception to this is possibly braille chording [7, 8, 9] which have several current implementation for Smartphones such as mBraille and TypeInBraille [9]. Blind users already familiar with Braille can thus use such braille chording keyboards without having to learn a new chording alphabet.

Notably the most important developments during the last decades include the mobile device platform and the touch display. The mobile platform led to many new innovative strategies for inputting text without a full keyboard, most of which were based on keys. Research have focused on using numeric keypad-based input with and without letter disambiguation [10, 11] and devices with even fewer keys [12, 13]. Other input devices such as joysticks have also been studied [14].

The emergence of affordable touch displays sparked much research into virtual keyboards and gestures and currently touch display technology and mobile device technologies have converged into the mobile smart device platform.

Both research and the practice field shows that the classic QWERTY layout has survived despite its obvious inefficiencies. Many users are already familiar with the QWERTY layout from the desktop platform and thus do not have to learn a new text entry system. Recent research has thus instead focused on providing accurate text prediction to speed up the text input and abbreviation based input in languages with long compound words [15].

A challenge with the current state of the art mobile text entry, which is based on virtual keyboards with text prediction, is that the individual virtual keys are small and difficult to both see and hit. These text input interfaces are thus challenging to use for individuals with reduced motor abilities and reduced vision. In particular, older users find Smartphone text-entry difficult as they often have both reduced vision and motor functioning.

This paper takes a step back and explores Russian tap codes. To the best of the authors' knowledge Russian tap codes have not been explicitly explored in context of text input. Tap codes involve a simple system where letters are organized into a 5×5 grid where a letter is signaled by tapping the row and the column with a little pause in-between.

		Temporal (sequential actions)		
		One (T1)	Few (T2)	Many (T3)
Spatial (simultan-eous actions)	One (S1)			Morse code [16] Eye-blink [28]
	Few (S2)	Chording [5, 6] Joysticks [14]	T9-disambigu-ation [11] Mnemonic chording [20, 21]	Menu-based [13] Numpad-multitap
	Many (S3)	QWERTY-keyboard	QWERTY with prediction [3, 15]	Gestures [2]

Figure 1. Text entry design framework with impressionistic classification of text entry technologies from the literature.

Tap codes were typically used in prisons where inmates in different cells communicated with each other using the concrete walls. The success of the tap code, used without the aid of computers, was probably due to its simplicity and efficiency.

The tap code is comparatively much easier to learn than the widely used Morse code, which has shown to be less suitable for text entry [16]. Both Morse code and tap codes rely on rhythmic patterns, and it has been found that users naturally exhibit regular rhythmic patterns when entering text using certain systems [17].

Some of the research efforts and commercial products available share similarities with the tap codes, instead of taps two-dimensional scrolling is used to select the desired character [18]. The elegance of tap codes is that users do not hit a specific target, but rather signal the information in easy-to-understand and simple temporal pulses. Since no target hitting tasks are involved, issues of accuracy, speed and size described by Fitts' law [19] does not apply. This study discusses the tap codes in context of recent text entry research and current technology.

2. METHOD

First, a framework for classifying text entry designs is presented. This framework allows the strengths and weaknesses of a design to be identified easily, and it allows poor designs to be discarded before committing to time-consuming user testing. Next, the classic Russian tap code and several designs adaptions based on the classic tap code are analyzed using the framework.

2.1 Text Entry Design Classification

Text entry designs often balance coding in the temporal and spatial domain. The proposed classification framework therefore organizes text entry design according to these dimensions. Figure 1 shows an example of the classification framework applied to several text entry strategies reported in the literature.

The framework divides the temporal (T) and spatial (S) domains into one (1), few (2) and many (3). In this classification the classic QWERTY keyboard is classified as S3-T1, that is, each of the many characters have a unique spatial position on the keyboard (many) and only one step is needed to recall each character (T1). The classic chording techniques are classified as S2-T1 as only one step is needed and usually only five keys are used. Note that variations on the chording keyboard, such as one based on visual mnemonics, require several steps [20, 21] and are therefore classified as S2-T2.

The classic Tegic T9 disambiguation technique for numeric keypad input is also classified as S2-T2 as it involves only 8 keys (few) and character input with subsequent word selection. Gestures are classified as S3-T3 since a gesture in reality is a complex temporal and spatial shape, although the actual execution of a gesture may be simple.

1,1	1,2	1,3	1,4	1,5
A	B	CK	D	E
2,1	2,2	2,3	2,4	2,5
F	G	H	I	J
3,1	3,2	3,3	3,4	3,5
L	M	N	O	P
4,1	4,2	4,3	4,4	4,5
Q	R	S	T	U
5,1	5,2	5,3	5,4	5,5
V	W	X	Y	Z

1,1	1,2	1,3	1,4	1,5
A	B	C	D	E
2,1	2,2	2,3	2,4	2,5
F	G	H	IJ	K
3,1	3,2	3,3	3,4	3,5
L	M	N	O	P
4,1	4,2	4,3	4,4	4,5
Q	R	S	T	U
5,1	5,2	5,3	5,4	5,5
V	W	X	Y	Z

Figure 2. The classic tap-codes (left), Polybius square (right).

1,1	1,2	1,3	1,4	1,5
A	B	C	D	E
2,1	2,2	2,3	2,4	2,5
F	G	H	I	J
3,1	3,2	3,3	3,4	3,5
K	L	M	N	O
4,1	4,2	4,3	4,4	4,5
P	Q	R	S	T
5,1	5,2	5,3	5,4	5,5
U	V	W	X	Y

1,1	1,2	1,3	1,4	1,5
E	T	O	S	L
2,1	2,2	2,3	2,4	2,5
A	I	H	C	F
3,1	3,2	3,3	3,4	3,5
N	R	U	G	B
4,1	4,2	4,3	4,4	4,5
D	M	Y	V	J
5,1	5,2	5,3	5,4	5,5
W	P	K	X	Q

Figure 3. Enhanced tap codes: no-z layout (left), letter frequency optimized square (right) (right).

Note that the classic QWERTY keyboard also can be partially considered a chord keyboard as SHIFT is usually pressed together with a character to produce uppercase characters. It could also be placed in the T2 category when considering the use of CAPSLOCK needing two steps. However, for simplicity, these low-probability cases are ignored herein.

One interpretation of the framework is that increased spatial complexity requires pointing accuracy, while increased temporal complexity requires both physical and mental effort. These issues need to be carefully balanced for a given context. Note that it is no practical input strategy for S1-S2 as it would have to be based on a binary alphabet. Moreover, no existing text input strategy could be identified for S1-T2, that is, text input strategies that do not require hitting a specific target, and that require few input steps.

2.2 Classic Tap Codes

The classic Russian tap-codes were based on the Russian alphabet. However, the English version that was used in the American prisons comprised a 5 × 5 grid of characters (see Figure 2). Since English has 26 characters K is given the same code as C. To send a D the user first tap once to signal the first row, then a short pause, followed by four consecutive taps to signal the fourth character along the row. That is, D would be communicated using tap + pause + tap + tap + tap + tap.

Tap codes go back to the Polybius square that was used in ancient Greece where rows of flames were used to send signals according to row and column coding. The typical English version of the Polybius square places the character J in the same cell as I.

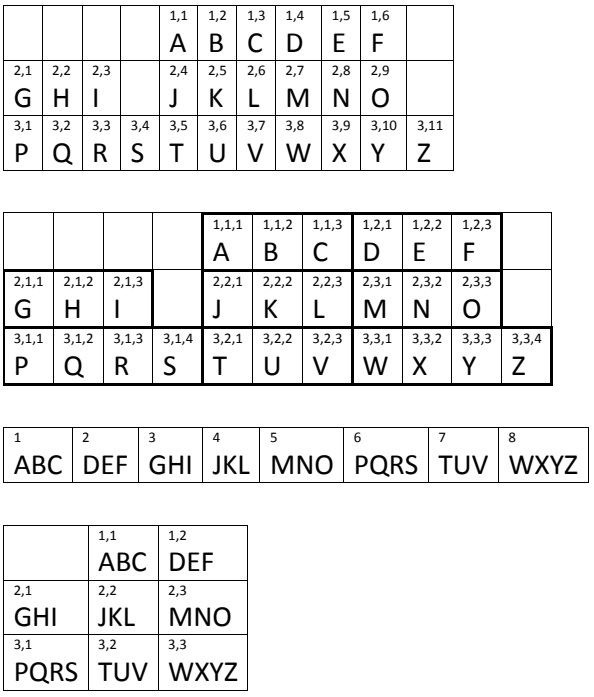

Figure 4. Numeric keypad-based tap-codes: multitap-layout (top), three-dimensional multitap (second from top), one-dimensional T9 (second from bottom), two-dimensional T9 (bottom).

The effectiveness of these classic tap-codes are explored herein using the proposed framework and the keystrokes per character (KSPC) measure proposed by MacKenzie et al. [22].

2.3 Enhanced Tap Codes

Three basic variations on the tap codes are investigated. First, the effects of starting in different corners are explored, that is, starting on the right instead of on the left, or on the bottom instead of on the top tapping along the opposite direction. Four cases are thus considered, top-left, top-right, bottom-left and bottom-right.

Next, Z is the character with the lowest frequency in English. Calculations were thus performed to explore the effect of giving K its own cell and instead discarding Z (Figure 3).

Finally, a tap-code optimized according to letter frequencies were explored (see Figure 3) where the most frequent characters were assigned the cells requiring the fewest taps. The goal was to determine the theoretically lower bound on the number of taps needed. However, research also show that such optimized layouts require new learning and hence are considered uncomfortable to use despite being the most efficient. The worst-case layout is also explored. It is simply the optimized layout mirrored along the diagonal going from the bottom left to the top right cell.

2.4 Numeric Keypad Layouts

Unfortunately, the alphabetic square layout requires learning [23, 24]. Instead, existing layouts already familiar to users are explored. One of these is the numeric keypad found on older mobile phones.

The first of four designs simply involves placing the characters out in a 3 × 11 grid resembling the actual character sequence of the numeric keypads (see Figure 4 top). The blank spaces are ignored.

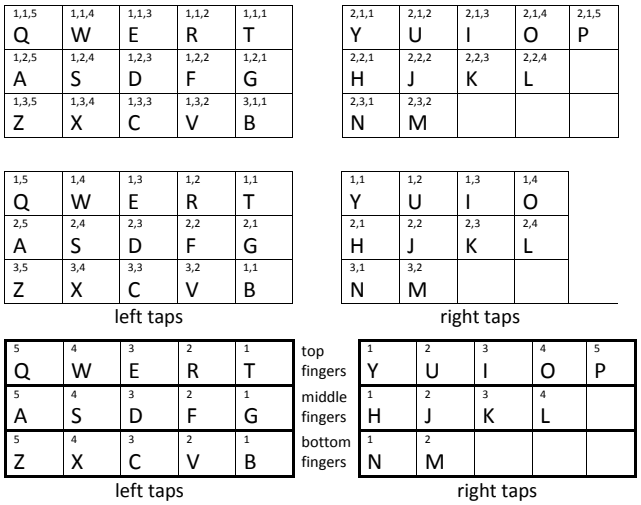

Figure 5. QWERTY-based tap codes: three-dimensional QWERTY tap code (top), two-hand two-dimensional tap code centered in the top middle (center) and two-hand three-finger one-dimensional QWERTY tap codes (bottom).

		Temporal (sequential actions) One (T1)	Few (T2)	Many (T3)
Spatial (simultaneous actions)	One (S1)		T9 grid Optimized tap-code	QWERTY, Russian, Polybian, numpad, worst-case tap code
	Few (S2)	QWERTY three-finger	QWERTY Bimanual	
	Many (S3)			

Figure 6. Tap-code designs overview.

The second design is three-dimensional (see Figure 4 second from the top). First, the user taps the desired row on the "keypad", next the user taps the desired column and finally taps to the desired character on the "key".

The last two designs in the numeric keypad category leans on numeric keypad input disambiguation where only the "button" with the set of characters is input, and a dictionary is used to disambiguate which word was intended. The first design is a simple one-dimensional tap code where the user taps through to the desired letter-group.

The second design is two-dimensional where the user selects the letter group by tapping the desired row followed by the column (see Figure 4 bottom).

2.5 Exploring QWERTY Familiarity

Finally, tap-codes based on the well-known QWERTY layout were explored. The first design is a three-dimensional tap code where one tap is first used to select the left side or two taps are used to select the right side. Next, one, two or three taps are used to select the row, and finally taps are used to select the desired column starting from the center (see Figure 5 top).

The following two designs are two-dimensional tap codes based on the assumption that the user can tap independently using the left and the right hand, and thereby control the selection for each side of the layout.

Table 1. Characteristics of the tap-code designs.

	taps				Characteristic			
Design name	Min	Max	mean	class	Dimension	Hands	Fingers	Dictionary
Russian (top left)	2	10	5.77	S1-T3	2D		1	
Russian (top right)	2	10	5.37	S1-T3	2D		1	
Russian (bottom left)	2	10	7.62	S1-T3	2D		1	
Russian (bottom right)	2	10	7.22	S1-T3	2D		1	
Polybian	2	10	5.79	S1-T3	2D		1	
No-z	2	10	6.10	S1-T3	2D		1	
Frequency optimized	2	10	4.48	S1-T3	2D		1	
Worst case	2	10	8.51	S1-T3	2D		1	
Numpad layout	2	14	6,70	S1-T3	2D		1	
Numpad cube	3	10	5.50	S1-T3	3D		1	
T9 linear	1	8	5.46	S1-T3	1D		1	Yes
T9 grid	2	6	4.84	S1-T2	2D		1	Yes
QWERTY	3	9	5.76	S1-T3	3D		1	
QWERTY bimanual center	2	8	4.35	S2-T2	2D	2	2	
QWERTY bimanual left	2	8	4.53	S2-T2	2D	2	2	
QWERTY three-finger	1	5	2.72	S2-T1	1.5D	2	6	

One variation starts in the top left corner for both sides, while the other variation starts in the middle, that is, in the top left corner for the left side and the top right corner for the right side. This design assumes that the skills of accessing the QWERTY layout mirrors across hands [25].

The final QWERTY based tap code assumes that three fingers on each hand are used to tap the keys associated with the respective row of the keyboard. This design thus constitutes six parallel tap codes. Simple pointing is needed to place fingers in the initial position, but it is similar to chording in that this is only done once when starting.

These last three QWERTY designs deviate from the classic one-channel tap code paradigm by allowing tapping on different channels. This is not possible with prison walls, but this prison wall limitation obviously does not apply to computers.

3. RESULTS

The results of the analysis are discussed according the three layout categories, namely alphabet square layouts, numeric keypad layouts and QWERTY layouts.

3.1 Alphabet Square Layouts

Figure 6 shows that most of the two dimensional tap codes belong in the S1-T2 category. The classic Russian tap-code clearly requires a minimum of two taps and a maximum of 10 taps, with an average of 5.7 taps per character. The starting point and tapping direction clearly have a significant effect on the number of taps. Luckily, the classic tap-code configuration is just marginally worse than the theoretically best which is achieved by starting in the top right corner with a mean of 5.37 taps per character. When starting at either the left or right bottom corner the mean taps per characters exceed 7.

The English version of the Polybius square gives only a marginally lower mean of 5.79 taps per characters. However, reshuffling the characters of the grid by removing Z has a larger impact and results in the mean number of taps per character exceeding 6.1.

The results for the frequency optimized tap code gives a mean of 4.48 taps per character, which is better than the T9 grid. The disadvantage with the optimized grid is that the user will have to learn a new unintuitive layout. However, no dictionary is used and any character sequence can therefore be input. The worst-case layout reveals an upper bound of a mean of 8.51 taps per character.

Common for all the tap code strategies based on square layouts is that a large number of taps is needed to enter characters regardless of layout, and at most 10 taps are needed to produce the worst-case character. Clearly, one does not need to conduct a user evaluation to predict that the amount of tapping is time-consuming, repetitive and frustrating for users. Even the input of simple phrases require many taps. In prisons the tap codes were used with short form coding such as GBU – God bless You, etc. Therefore, tap codes could perhaps be combined with some intuitive abbreviation expansion engine [15] or telegram expansion techniques inspired by SOUNDEX [26] or Metaphone [27]. For such mechanisms to succeed they have to be simple and require minimal learning. Although tap codes in its classic form is not the optimal text input strategy for most users given modern hardware, it may be a workable solutions for individuals with certain forms of reduced motor functioning. Taps do not necessarily have to be performed with fingers per se, it could equally well be another modality such as eye-blinking [28], non-finger physical movement, an uttered sound, etc.

One important characteristic to be learnt from the tap codes is the square configuration as this is the optimal two-dimensional layout in terms of character access and effort needed. However, several commercial designs involving alphabetic layout have chosen non-square rectangular layout accessible via navigation buttons. The non-square configurations increases the average number of steps to

access a given character. The idea of combining two characters such as C and K with the Russian square or I and J with the Polybius square could also be adopted in regular alphabetic layouts to achieve square layouts. Combining two characters with relatively low probability will be trivial to disambiguate using a simple language model. Other European languages with extended alphabets could also be reduced to a simple 5 × 5 square of the sake of simplicity. Previous keyboard layout research [4] has focused more on the position of the characters than on the actual shape of the keyboard layouts.

3.2 Numeric Keypad Layouts

The results of the evaluation shown in Figure 6 reveal that the T9 grid approach is classified as S1-T2, a category that is not documented in the text entry research literature. This means that its advantage is single channel of taps with few symbols. Table 1 shows that the average number of taps per character with T9 grid is 4.48 with a maximum of 6 taps. The disadvantage of T9 grid is that it requires disambiguation and it is thus impossible to input tokens not present in dictionaries. Moreover, the statistics does not take the extra manual disambiguation steps into consideration, that is, when the user have to select between different options.

Although the numeric layout is familiar to most users, the analysis shows that it does not pose any particular advantage when physical effort is considered together with the mental effort needed. Moreover, one may ponder whether people's familiarity with the numeric keypad layout is disappearing as the classic T9 mobile phones used for texting have been replaced with touchscreen Smartphones. The designs based on the numeric keypad layouts can thus be discarded.

3.3 QWERTY Layouts

Two strategies that stand out in category S2-T1 and S2-T2 are QWERTY three finger and QWERTY bimanual, respectively. Of the two bimanual methods, the one based on starting in the center gives a slightly better mean of 4.35 taps per character over 4.53 by always starting in the top left corner. These results are better than both T9 grid and the optimized square. Moreover, it utilizes a layout already familiar to the user and does not involve any disambiguation. However, it relies on the user being able to use two channels of tapping, preferably using the left and the right hand to mimic keyboard based bimanual text entry.

The three-finger tap code is also a bimanual text input strategy and actually involves six fingers in total (3 for each hand). This strategy yields the lowest mean taps per keystroke, namely 2.72, with a minimum of 1 and maximum of 5 taps. It is mentally simple to use since it is a one dimensional tap code, and no pauses are needed to enter characters. It relies on the familiar QWERTY layout and users already familiar with QWERTY will therefore not need to search for the location of the characters. However, six fingers are needed to tap on six channels without pointing tasks. This require more dexterity than single channel tapping. However, if one assumes that the fingers remain stationary on the control keys as no pointing and hitting of targets are needed.

Overall, based on the results future work should focus on the three text entry strategies based on the QWERTY layout. A natural subsequent step is to conduct a user evaluation in the form of a text entry experiment to evaluate whether the three methods are feasible in practice. Moreover, future work should involve experimentation involving users with motor impairments to assess whether the methods are beneficial to these users or not.

4. CONCLUSIONS

The classic tap code, or knock code, was explored in light of recent research into text entry and modern input devices. Tap codes are attractive as no pointing tasks are needed. Moreover, a classification framework was proposed for exploring and screening text entry designs allowing poor designs to be discarded prior to time-consuming user testing. Note that the framework is not intended as a replacement for user testing. The results show that the classic tap codes used in prisons are relatively efficient compared to related designs. However, acceptance for the tap code may be improved if the familiar QWERTY layout is used instead. The results show that the efficiency of tap codes based on the QWERTY layout is comparable to that of the classic alphabetical square. However, the QWERTY layout also allows easy facilitation of bimanual tapping and multi-finger tapping. These optimizations greatly reduce the length of the tedious tap sequences without increasing complexity. Future research need to confirm the feasibility of the derived designs through user evaluations. The results of this study also suggest that text entry systems based on selecting letters using navigation keys from an alphabetical layout also should employ a square grid instead of rectangular layouts to minimize motor effort.

5. REFERENCES

[1] François Portet, Michel Vacher, Caroline Golanski, Camille Roux, and Brigitte Meillon. 2013. Design and evaluation of a smart home voice interface for the elderly: acceptability and objection aspects. Personal Ubiquitous Comput. 17, 1 (January 2013), 127-144. DOI=http://dx.doi.org/10.1007/s00779-011-0470-5

[2] Frode Eika Sandnes, Tek Beng Tan, Anders Johansen, Edvin Sulic, Eirik Vesterhus, and Eirik Rud Iversen. 2012. Making touch-based kiosks accessible to blind users through simple gestures. Univers. Access Inf. Soc. 11, 4 (November 2012), 421-431. DOI=http://dx.doi.org/10.1007/s10209-011-0258-4

[3] John J. Darragh, Ian H. Witten, and Mark L. James. 1990. The reactive keyboard: A predictive typing aid. Computer 23, 11 (1990), 41-49.

[4] I. Scott MacKenzie and Shawn X. Zhang. 1999. The design and evaluation of a high-performance soft keyboard. In Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI '99). ACM, New York, NY, USA, 25-31. DOI=http://dx.doi.org/10.1145/302979.302983

[5] Nathaniel Rochester, Frank C. Bequaert, and Elmer M. Sharp. 1978. The chord keyboard. Computer 11, 12 (1978), 57-63.

[6] Daniel Gopher and David Raij. 1988. Typing with a two-hand chord keyboard: will the QWERTY become obsolete? IEEE Transactions on Systems, Man and Cybernetics 18, 4 (1988), 601-609.

[7] David A Fisher and C. Ward Bond. 1992. A single-handed braille chord system for computer keyboard input. In Proceedings of Computing Applications to Assist Persons with Disabilities, 1992. IEEE, 200-202.

[8] Myung-Chul Cho, Kwang-Hyun Park, Soon-Hyuk Hong, Jae Wook Jeon, Sung Il Lee, Hyuckyeol Choi, and Hoo-Gon Choi. 2002. A pair of braille-based chord gloves. In Proceedings of the Sixth International Symposium on Wearable Computers 2002 (ISWC 2002). IEEE, 154-155.

[9] Sergio Mascetti, Cristian Bernareggi, and Matteo Belotti. 2011. TypeInBraille: a braille-based typing application for

touchscreen devices. In The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility (ASSETS '11). ACM, New York, NY, USA, 295-296. DOI=http://dx.doi.org/10.1145/2049536.2049614

[10] I. Scott MacKenzie, Hedy Kober, Derek Smith, Terry Jones, and Eugene Skepner. 2001. LetterWise: prefix-based disambiguation for mobile text input. In Proceedings of the 14th annual ACM symposium on User interface software and technology (UIST '01). ACM, New York, NY, USA, 111-120. DOI=http://dx.doi.org/10.1145/502348.502365

[11] Jun Gong, Bryan Haggerty, and Peter Tarasewich. 2005. An enhanced multitap text entry method with predictive next-letter highlighting. In CHI '05 Extended Abstracts on Human Factors in Computing Systems (CHI EA '05). ACM, New York, NY, USA, 1399-1402. DOI=http://dx.doi.org/10.1145/1056808.1056926

[12] Scott MacKenzie. 2002. Mobile text entry using three keys. In Proceedings of the second Nordic conference on Human-computer interaction (NordiCHI '02). ACM, New York, NY, USA, 27-34. DOI=http://dx.doi.org/10.1145/572020.572025

[13] Frode Eika Sandnes, Haavard W. Thorkildssen, Alexander Arvei, and J. O. Buverad. 2004. Techniques for fast and easy mobile text-entry with three-keys. In Proceedings of the 37th Annual Hawaii International Conference on System Sciences 2004. IEEE.

[14] Frode Eika Sandnes and Andre Aubert. 2007. Bimanual text entry using game controllers: Relying on users' spatial familiarity with QWERTY. Interact. Comput. 19, 2 (March 2007), 140-150. DOI=http://dx.doi.org/10.1016/j.intcom.2006.08.003

[15] Frode Eika Sandnes. 2015. Reflective Text Entry: A Simple Low Effort Predictive Input Method Based on Flexible Abbreviations. Procedia Computer Science 67 (2015), 105-112.

[16] Simon Levine, John Gauger, Lisa Bowers, and Karen Khan. 1986. A comparison of Mouthstick and Morse code text inputs. Augmentative and Alternative Communication 2, 2 (1986), 51-55.

[17] Frode Eika Sandnes and Hua-Li Jian. 2004. Pair-wise variability index: Evaluating the cognitive difficulty of using mobile text entry systems. In Proceedings of the International Conference on Mobile Human-Computer Interaction, Lecture Notes on Computer Science 3160, Springer Berlin Heidelberg, 347-350.

[18] Didier Augusto Vega-Oliveros, Diogo de Carvalho Pedrosa, Maria da Graça Campos Pimentel, and Renata Pontin de Mattos Fortes. 2010. An approach based on multiple text input modes for interactive digital TV applications. In Proceedings of the 28th ACM International Conference on Design of Communication (SIGDOC '10). ACM, New York, NY, USA, 191-198. DOI=http://dx.doi.org/10.1145/1878450.1878483

[19] I. Scott MacKenzie, Shawn X. Zhang, and R. William Soukoreff. 1999. Text entry using soft keyboards. Behaviour & information technology 18, 4 (1999), 235-244.

[20] Frode Eika Sandnes. 2015. Human performance characteristics of three-finger chord sequences. Procedia Manufacturing 3 (2015),.4228-4235.

[21] Frode Eika Sandnes. 2006. Can spatial mnemonics accelerate the learning of text input chords?. In Proceedings of the working conference on Advanced visual interfaces (AVI '06). ACM, New York, NY, USA, 245-249. DOI=http://dx.doi.org/10.1145/1133265.1133313

[22] I. Scott MacKenzie. 2002. KSPC (keystrokes per character) as a characteristic of text entry techniques. In Proceedings of Human Computer Interaction with Mobile Devices. Springer Berlin Heidelberg, 195-210.

[23] Donald A. Norman and Diane Fisher. 1982. Why alphabetic keyboards are not easy to use: Keyboard layout doesn't much matter. Human Factors: The Journal of the Human Factors and Ergonomics Society 24, 5 (1982), 509-519.

[24] Frode Eika Sandnes. 2010. Effects of common keyboard layouts on physical effort: Implications for kiosks and Internet banking. In Proceedings of Unitech2010: International Conference on Universal Technologies, Tapir, Trondheim, Norway, 91-100.

[25] Edgar Matias, I. Scott MacKenzie, and William Buxton. 1994. Half-QWERTY: typing with one hand using your two-handed skills. In Conference Companion on Human Factors in Computing Systems (CHI '94), Catherine Plaisant (Ed.). ACM, New York, NY, USA, 51-52. DOI=http://dx.doi.org/10.1145/259963.260024

[26] David Holmes and M. Catherine McCabe. 2002. Improving precision and recall for soundex retrieval. In Proceedings of the International Conference on Information Technology: Coding and Computing, 2002. IEEE, 22-26.

[27] Lawrence Philips. 1990. Hanging on the metaphone. Computer Language 7, 12 (December, 1990), 39-43.

[28] Aleksandra Królak and Paweł Strumiłło. 2012. Eye-blink detection system for human–computer interaction. Universal Access in the Information Society 11, 4 (2012), 409-419.