Single-Handed Typing with Minimal Eye Commitment: A Text-Entry Study

Adrian Tarniceriu, Pierre Dillenbourg, Bixio Rimoldi
School of Computer and Communication Sciences
Ecole Polytechnique Federale de Lausanne
Lausanne, Switzerland
adrian.tarniceriu@epfl.ch, pierre.dillenbourg@epfl.ch, bixio.rimoldi@epfl.ch

***Abstract*—For most people, typing text on a mobile device requires visual commitment to the input mechanism. As a consequence, there are many situations in our daily life when we have to refrain from using these devices as our vision is already committed. An example is trying to text while walking in a crowded place. Chording devices allow us to type without looking at the input device but using them requires some training. We present the results of a study that evaluates the performance of a key-to-character mapping for a 5-key chording device. The mapping is designed to minimize the learning phase. After 45 minutes of training it was completely learned, and after approximately 250 minutes the average entry speed was 15.2 words per minute. A prototype that implements this mapping was mounted on a bike and tested by the authors who could comfortably ride and type while being focused on the road.**

Keywords-chording keyboard; text entry; key mapping; mobile device

I. Introduction

Mobile computing devices such as smart phones or personal digital assistants play an ever increasing role in our daily lives. More and more people appreciate their services and want to have access to them at all times, but their input methods are not suitable during activities for which vision is partially or entirely required. For example, in a crowded place most people need to stop walking in order to text via today's keypads or touchscreen keyboards.

Chording is a text-input method that utilizes a small number of keys. A character is formed by pressing a key combination, similarly to playing a chord on a musical instrument. With five keys, one for each finger, we have 31 different combinations in which at least one key is pressed. This is enough for the 26 letters of the English alphabet and five punctuation signs. If we can keep our fingers on the keys, then we can type with one hand and without looking at the keys. Our vision (or auditive feedback) is still needed occasionally to verify the output, but this requires considerably less commitment than continuously looking at the input device. Visual commitment can be further reduced by displaying the output in the natural field of vision, for instance on a windshield or on goggles.

The likely reason chording devices are not popular is that users require some training before being able to type. Compared to a QWERTY keyboard, where users can hunt and peck from the beginning, the mapping between keys and characters has to be learned for chording devices. The overhead needed to do so depends on the keyboard type and mapping and can vary by several hours. The main objective of this paper is to present the results of a study on a mapping designed to facilitate the learning task.

The paper is organized as follows. In Section II, we present a brief overview of related work. In Section III, we describe a key-to-character mapping for a 5-key keyboard designed to reduce the learning time. We denote this mapping in the following as 5keys. In Sections IV and V, we present two experiments that evaluate the learnability, text-entry rates, typing accuracy and common error patterns. In Section VI, we conclude the paper and discuss future directions.

II. Related Work

The first chording applications were used in stenotype machines (1830's), telegraph communications, and the Braille system that allows blind people to read and write.

Subsequent studies were performed by IBM [1], but the results were considered inconclusive and research was stopped in 1978. Douglas Engelbart, the inventor of the computer mouse also proposed a 5-key keyset, but this was not incorporated in any system [2], probably due to the popularity of the standard QWERTY keyboard.

As mobile devices and wearable technologies evolve, classic keyboards are no longer able to fulfill users' needs for mobility and ubiquitous access to computational resources. Therefore, chording keyboards have re-emerged as a popular research topic, leading to the appearance of several devices, like DataEgg [3], GKOS [4], Twiddler [5], EkaPad [6] and the chording glove [7]. Depending on the envisaged application, these keyboards can have different number of keys, mappings or shapes, each of these being a research topic. In the following, we will focus on a 5-key keyboard and a mapping designed to minimize the learning process.

III. Character Mapping

An important aspect of designing a chording keyboard is the mapping between key combinations and desired characters. One possibility is to assign easier combinations for more frequent letters, as in the Morse code, leading

 ISBN: 978-1-61208-236-3

to higher typing speeds. Even if these mappings are easy to determine, the user must learn by heart the key-to-letter correspondence as there is no intuitive link between them. Another possibility is to use a semantically richer mapping, which would be easier to learn.

The key-to-character mapping studied in this work was designed with the primary goal of making it easy to remember. It was designed for a 5-key keyboard, where each character is represented by a different key combination. In this paper we will focus only on lowercase letters plus the period, space and backspace, as they are the most used; also because most typing studies consider only this set of characters.

To create enough possibilities for assigning an intuitive key combination to each character, we conceived five mnemonic categories.

1) Single-key category: remembering the map for the characters in this category should be totally trivial. Characters are produced by pressing a single finger and the letter is the initial of the finger. So by pressing the key under **t**humb, **i**ndex, **r**ing and **p**inky, we obtain "**t**", "**i**", "**r**", and "**p**", respectively. There is an exception to the rule: since "**m**" fits well in another category (see below), we have reserved the middle finger for the period.
2) Fingers-down category: the most natural way to produce the shape of an "**m**" with the hand is to stretch down the index, middle, and ring fingers (see Figure 1b). In a similar fashion we can produce the other letters in this category, namely "**n**", "**u**", "**y**", and "**c**".
3) Fingers-up category: similarly, a natural way to produce the shape of a "**w**" is to stretch up the index, middle, and ring fingers (Figure 1c). The associated character is obtained by pressing the key(s) under the remaining fingers. "**v**", "**l**", "**e**", and "**j**", follow the same idea. We have included **space** and **backspace** in this category as backspace can be associated with the thumb pointing to the left and space with the pinky pointing to the right.
4) Finger footprint category: usually, a character is produced faster with a 5-key device if the users first think of the fingers pressing the keys and not of the fingers that remain "up". Considering this, for "**h**" we can identify three landmark spots on the shape of the letter and associate them to fingers according to the following rule: the thumb is for spots that are left and low, the index for left high, the ring for right high, and the pinky for right low. The resulting mapping is given in Figure 1d. With a little bit of imagination in this category we can also fit "**a**", "**f**", "**k**", "**o**", "**s**", "**x**", and "**z**". For "**o**", we imagine five dots spread around a circle, and we obtain it by pressing all buttons.
5) Associative category: the letters "**b**" and "**d**" may be seen as an "**o**" with a vertical bar on the left and right, respectively. We use the index and the ring fingers to represent these bars. "**g**" was inspired from "**y**" (they look alike in handwriting) and in turn "**g**" inspires "**q**" (the tail ends left and right respectively, so for "**g**" we use the thumb and for "**q**" the pinky).

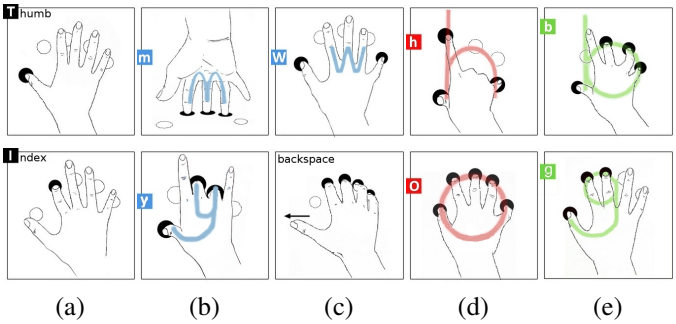

Figure 1. Examples for letter mappings. (a) Single-key mnemonics for "**t**"and "**i**". (b) Fingers-down mnemonics for "**m**" and "**y**". (c) Fingers-up mnemonics for "**w**" and "**backspace**". (d) Finger footprint mnemonics for "**h**" and "**o**". (e) Associative mnemonics for "**b**" and "**g**"

Two examples from each category are given in Figure 1.

Some of the above mnemonics are easier to remember than others. With five keys, however, there are only 31 usable combinations and we use them all to map the 26 characters plus the space, backspace, period, enter and comma. Hence any change aimed at improving one mnemonic implies at least one other change.

The effectiveness of the proposed mapping is assessed through two experiments described in the next sections. The first compares this mapping to two others from a learnability point of view, or, in other words, how easy users can remember the key-to-character correspondences. The second experiment estimates the usability of the mapping (typing speed, accuracy [8] and most common mistakes).

IV. Learnability Study

A. Experimental Setup

This first experiment compares, from a learnability point of view, the proposed mapping (5keys) to two others. The references are the Microwriter mapping [9], also based on intuitive mnemonics, and the Baudot code [10] that is based on letter frequency and assigns easier key combinations to most common characters. All three mappings are designed for 5-key keyboards.

A Java application was designed to simulate the chording keyboard on a regular QWERTY desktop keyboard. It only allows the use of five keys, each representing a key of the chording keyboard. Each of these keys correspond to a finger of the right hand. A typical choice was the spacebar for the thumb, and the keys for f, t, y and u for the index, middle finger, ring and pinky, respectively.

The experiment consisted of three sessions of three rounds each. For each round, the subjects had 5 minutes to look

 ISBN: 978-1-61208-236-3

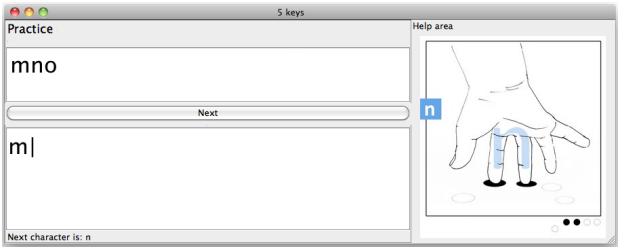

Figure 2. Typing application screenshot

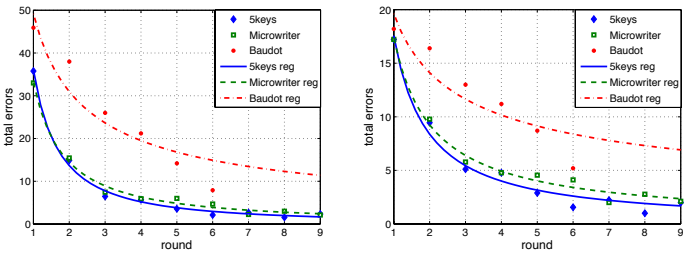

Figure 3. (a) Average number of errors (for each mapping and for each round) and regression curves. (b) Average number of character errors (for each mapping and for each round) and regression curves

at a printed version of the mappings and try to remember them. Afterwards, they used the Java application to warm up by typing each letter of the alphabet. A help image showing the key combination for the letter to be typed was shown to the participants. A screenshot of the application is visible in Figure 2. The top-left window contains the target characters to be typed. The bottom-left window represents the typing area and the help image is displayed on the right. In the next step, the help image was not available any more and the participants had to type the alphabet three times. The order of the letters was random, but the same for all participants. The subjects had five seconds and only one attempt to type each target character. The correct key combination was displayed when the user typed a character (right or wrong) or when the five seconds expired.

30 participants, 10 for each of the three mappings, were recruited from the students of our university (undergraduate, master and PhD programs). They were between 19 and 30 years old, and four were female. For each session of the experiment (approximately 30 minutes) they received a fixed monetary compensation. None of the subjects had used a chording keyboard before. As the participants who know how to play a musical instrument could have had an advantage, they were equally distributed among the three experiment groups. We also tried to equally distribute them based on gender and study level. Two participants abandoned the experiment after the first session, one testing the 5keys and one the Microwriter mapping.

B. Experiment Results

To determine which of the mappings is easier to learn we compared the number of errors (wrongly typed or not typed characters) for each round. Exponential regressions [11] were derived to fit these error values. The average values for each mapping and for each round and the exponential regressions are presented in Figure 3a.

After two sessions (six rounds of approximately five minutes of typing each), the total number of errors was considerably lower for the mnemonic based mappings (5keys and Microwriter) compared to the mapping based on letter frequency. Therefore, we conclude that mnemonic based mappings are learned faster. The goal of the study was to evaluate which mapping is easier to learn and the Baudot mapping is clearly more difficult. Hence, in the third session we only analyzed the 5keys and Microwriter mappings. By checking only the average values, no significant difference between these two was noticed. An advantage of 5keys can be observed from the analysis of the regression curves as the curve for 5keys is slightly below the curve for Microwriter.

Besides the total number of errors, we also compared the number of wrong characters per round. For example, if "**a**" was typed incorrectly two times, this counts only as one character error. The results are shown in Figure 3b. In this case the difference between 5keys and Microwriter mappings is more visible and, as expected, both lead to considerably less errors than the Baudot mapping.

At the end of the third typing session (after nine rounds or approximately 45 minutes of actual typing), the participants were asked how confident they feel about their knowledge of the mappings and if they could use the presented method as a text input mechanism. All of them answered affirmatively and most also mentioned that they completely learned the mappings. This is confirmed by a low error rate (3.16% after 6 rounds and 2.14% after 9 rounds for the proposed mapping).

From this experiment, we draw the conclusion that a mnemonic-based mapping facilitates the process of learning the code. We also conclude that the proposed 5keys mapping outperforms the Microwriter mapping, also mnemonic-based, in terms of average error rate.

The mnemonic set was designed based on the finger positions of the right hand. Two of the participants (one for the 5keys and one for the Microwriter) were left-handed. Yet they also typed with their right hand and, interestingly, their error rates were actually lower than the average.

V. Usability Study

The first experiment, aimed at evaluating the learning process, was followed by an independent experiment aimed at determining achievable typing rates, accuracy, and common error patterns for the 5keys mapping.

 ISBN: 978-1-61208-236-3

A. Experimental Setup

The second experiment was based on a similar Java application. This time, the subjects were asked to type full sentences, not just isolated letters. The experiment consisted of 11 sessions. For each of them, the subjects started by typing each letter from each category and continued with real sentences chosen from a set considered representative for the English language [12]. The first four sessions, of 25 minutes each, were for the participants to learn the mapping and to familiarize themselves with the keyboard. During these sessions the help image was always displayed. During the following sessions (5 to 10), each lasting 20 minutes, the help image was no longer available. Session 11 was similar, but only lasted for 10 minutes. For the first 10 sessions, the participants received a fixed monetary compensation. As an incentive, for the last one the reward was proportional to the subject's performance, measured by the number of correctly typed words. We recruited a new set of six students for this study. The number of participants is lower than for the first experiment due to the significant time commitment required.

In order to monitor the evolution of the experiment and to gather statistics about the subjects' activities and performances, for each session and for each participant the application generated several log files. These files recorded the typed text, the number of occurrences for each character, the corresponding key combination, the total number of errors, the number of corrected errors and the total time spent writing each character. When a typing error occurred, we checked what character was typed in lieu of the correct one.

B. Text-Entry Speed

We used the wpm (words-per-minute) measure to describe the text entry speed. This is defined as

$$wpm = \frac{60L}{t}\frac{1}{5} \qquad (1)$$

where L is the total number of typed characters and t is the typing time in seconds. The scaling factor of 1/5 is based on the fact that the average English word length is approximately 5 characters. As the average word length for the typed text differed from one session to another, the use of the above formula provides a more reliable estimate than actually counting the words.

The average entry rate for the first session was 4.2 wpm and reached 15.2 by the end of the experiment (after approximately 250 minutes of typing). Even though these values were obtained using five keys from a classic keyboard, they do give an estimate of what can be achieved using a real implementation of the device (the shape of the hand when the fingers are placed on the buttons is almost the same for a flat surface, bike handlebar, or around a mobile phone case).

Figure 4 presents the entry rates (for each subject, average and exponential regression) for each session. We observe

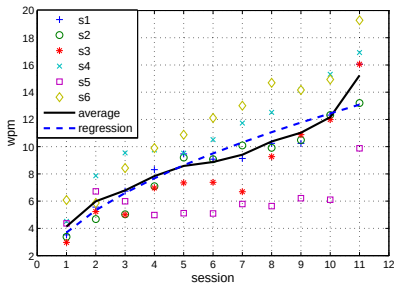

Figure 4. Typing rates per session for each user, average and regression

that the typing rates significantly improved from the 10th to the 11th session. This could be explained by the fact that for the last session the subjects were stimulated by a reward proportional to the number of correctly typed words.

As a reference, the typing rates achieved after 250 minutes of practice are 12.4 wpm for multi-tap mobile phones [13] and 20.6 wpm for Twiddler [5]. Rates of 20.36 wpm were reached by expert T9 users [14]. We should point out that the experimental conditions were not the same for all devices. Hence, the above typing rates are only of indicative nature. For both multi-tap and T9 techniques visual attention is essential for most users. For the 5keys device it makes essentially no difference if the user has visual contact with the keys or not. It should also be taken into consideration that Twiddler uses 12 keys, whereas our mapping only requires 5 keys, thus providing a clear space advantage and more design flexibility. If placed in a position which is naturally under the fingertips (for example on the handlebar of a bike), the users will have continuous access to the keys.

C. Error Analysis

Starting with session 5 (when the help image was no longer displayed) we evaluated the accuracy based on the corrected and uncorrected errors. The error percentages are defined as

$$corrected\% = \frac{\#backspaces}{\#characters}100\,, \qquad (2)$$

$$uncorrected\% = \frac{\#incorrect\_characters}{\#characters}100\,. \qquad (3)$$

The errors could have two main causes: the subject does not recall the correct key combination or, alternatively, a coordination mistake is produced during execution. We call these error types cognitive and sensorimotor errors, respectively. We expect the cognitive errors to decrease faster, as a function of training, because it is easier to learn the code than to improve motor skills. This is confirmed by the statements of the participants in both experiments: they said that they had learned the mapping by the end of the training, and errors were due to lack of attention or finger combinations that seemed difficult.

 ISBN: 978-1-61208-236-3

Table I
ERROR RATES PER SESSION

Session number	5	6	7	8	9	10
Corrected errors %	8.04	6.07	7.35	7.51	7.59	6.21
Uncorrected errors %	0.52	0.47	0.19	0.36	0.21	0.22

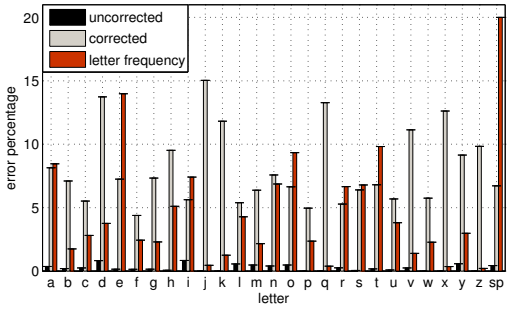

Figure 5. Error rates per character

Table I presents the error rates for sessions 5 to 10. For session 10 the uncorrected error rate is 0.2% as the subjects tended to correct most errors. The low uncorrected-error rate shows that the mapping was learned already by the end of session 4, as expected given the results of the first experiment.

Figure 5 presents the percentage of corrected and uncorrected errors per character for all users for the whole experiment. The figure also shows the scaled occurrence ratio for each character. Error rates are higher for less frequent letters (for example "**j**" and "**q**"), probably because the subjects had fewer opportunities to practice on them. Non-negligible error rates can also be observed for high-frequency characters, as users probably try to type faster as they gain more experience.

As in Section V-B, we compare the uncorrected error rates with those for multi-tap (5%) and Twiddler (3%) after 250 minutes of practice, and also with expert T9 users (0.52%). Even if the T9 entry method and Twiddler allow for higher typing rates, the error rates are also higher (by one order of magnitude for Twiddler). Again, these values are only indicative, due to different experimental conditions.

We also determined the dependency of the average error rate on the character category and on the number of keys that need to be pressed to compose a character. We observed that the error rates increase for characters that involve a larger number of keys, but the differences are not statistically significant (anova test p-values higher than 0.05). Letters from the single-key category have the lowest error rates and those from the associative category have the highest, but again, these results are not statistically significant.

D. Common Errors

To understand the error patterns that appear most frequently, we computed the confusion matrix [15] corresponding to the typed text. This is a square matrix with rows and columns labeled with all possible characters. The value at position *ij* shows the frequency of character *j* being typed when *i* was intended. The values are given as percentages from the total number of occurrences for character *i*.

It is useful to represent a key combination by a 5-bit codeword in which the first digit represents the key under the thumb, the second digit the key under the index, etc. The value of a position is 1 if the corresponding key is pressed. So, for instance, 11011 is the codeword for "**x**", for which all fingers except the middle one press the keys. By analyzing the 5-bit code for the 10 most common substitutions, we notice that in 9 of the 10 cases the errors appear between characters that differ only by one bit (for example "**x**", code 11011 and "**h**", code 11001). In the other case, two single-key characters are substituted ("**t**" and "**i**").

If we check word by word and consider only substitution errors, i.e., errors that arise from substituting individual characters, 84% of the erroneous words contain one substitution, 13% contain two substitutions and 2% three substitutions. From a bit-error point of view, 51% of the erroneous words contain a one-bit error, 33% a two-bit error and 14% a three-bit error. One-bit errors occur when the user does not press one of the required keys (31% of the total errors) or presses an extra key (20% of the total errors). Most two-bit errors are substitutions, when a wrong key is pressed and a correct key not pressed. These values can be used to implement an error correcting mechanism that relies both on a dictionary and on the probability that a character be substituted for another.

E. Character Typing Duration

As the coordination effort is not the same for all key combinations, we expect that different characters require more time than others to be typed. Figure 6 shows the average time per key combination for sessions 5 and 10. The time needed to form a key combination, called composition time, is measured from the moment the first key of a combination is pressed until a key is released. It is when a key of the combination is released that the corresponding character is produced. From that moment on, the pressing of a key indicates the start of a new character. Instead of ordering the letters alphabetically, in Figure 6 we order them in increasing number of pressed keys (single-key letters are first and "**o**", for which all keys are pressed, is last). The letters containing the same number of keys are ordered in ascending composition time for session 10.

As expected, the composition time increases with the number of pressed keys, the dependence being statistically significant (anova test p-values lower than 0.05). We also notice that letters requiring key combinations perceived as more difficult (for example "**q**", code 01101 or "**d**", code 11101 for which the middle finger and the pinky are down while the ring finger is up) require more time than others.

 ISBN: 978-1-61208-236-3

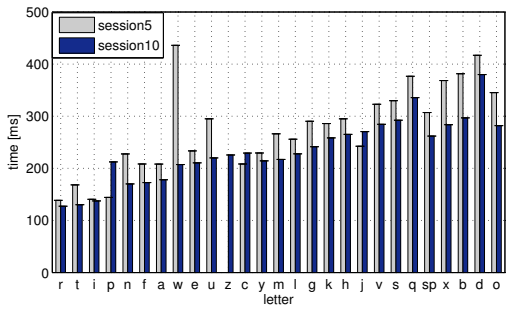

Figure 6. Average composition time per character for sessions 5 and 10

As subjects gained more experience, they were able to type faster and the average letter duration decreased from 273.9 milliseconds in session 5 to 234.5 in session 10, or by 14%. During the same period, text entry rates increased from 8.6 to 12.2 wpm, or by 41%. The difference is explained by the fact that the idle time between the end of one character and the beginning of the next also decreased.

VI. Conclusions and Future Work

In this paper, we have presented the results of a study evaluating the mapping for a chording input device. The overhead needed to learn the mapping was reduced by choosing easy-to-remember key combinations. A first experiment showed that the mapping was learned after less than 45 minutes of actual typing. Moreover, the total number of errors was considerably smaller than for a letter-frequency based mapping and slightly smaller than for another mnemonic based mapping.

A second experiment showed that after approximately 250 minutes of typing, the average text entry rate was 15.2 wpm and the uncorrected error rate 0.2%. We also analyzed which characters are perceived as more difficult to type and the most common errors. This data will be used to develop an error correction mechanism specifically designed for a chording keyboard using the proposed mapping. It will take into account a language model, as well as the probability that one character is typed for another.

During the experiments, the subjects sat at a desk. To go one step further, we designed, built and tested a prototype for a bike. We easily fit the five keys under the natural position of the fingers on the handlebar. Two of the authors tested the device and found that they could effortlessly ride and type while staying focused on the road. The position of the keys allowed them to control the bike with both hands while typing. Moreover, as the keys were directly under the fingers, they could also type accurately on a bumpy road. Though encouraging, these results are exploratory and a more accurate study should be performed.

There are numerous potential applications for a 5-key input device. For instance, with the buttons around a phone one can input a text message while walking (a stop to proof-read before sending should suffice). By means of a wrapper application that captures the text, a user can control the operation of a smartphone: in the test on the bike, the authors could control the music, write a short note, interact with the map, etc. We can easily envision the potential benefit of a 5-key input device on the handlebar of a shopping cart. It could be used to browse or edit a shopping list on a PDA placed in the middle of the handlebar or to interact with the store's web site to check the availability and the location of an item. Another interesting application could be to have the keys on the side of a TV remote control. Although modern TV sets allow for Internet navigation, typing a URL and doing searches with a standard remote control can still be quite clumsy.

References

[1] F. C. Bequaert and N. Rochester , "Teaching typing on a chord keyboard," tech. rep., IBM Technical Report, 1977.

[2] D. C. Engelbart, "Design considerations for knowledge workshop terminals," in *Proceedings of the June 4-8, 1973, national computer conference and exposition*, AFIPS '73, pp. 221–227, ACM, 1973.

[3] http://www.xaphoon.com/dataegg/, July, 2012.

[4] http://gkos.com/, July, 2012.

[5] K. Lyons, T. Starner, D. Plaisted, J. Fusia, A. Lyons, A. Drew, and E. W. Looney, "Twiddler typing: one-handed chording text entry for mobile phones," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '04, (Vienna, Austria), pp. 671–678, ACM, 2004.

[6] http://www.ekatetra.com/products/ekapad.html, July, 2012.

[7] R. Rosenberg and M. Slater, "The chording glove: a glove-based text input device," *IEEE Trans Syst, Man and Cybernetics, Part C: Applic and Rev*, pp. 186–191, 1999.

[8] R. W. Soukoreff, "Text entry for mobile systems: Models, measures, and analyses for text entry research," Master's thesis, York University, 2002.

[9] http://www.ericlindsay.com/palmtop/mwrite.htm, July, 2012.

[10] A. Ralston and E. D. Reilly, eds., *Encyclopedia of computer science (3rd ed.)*. Van Nostrand Reinhold Co., 1993.

[11] W. K. Estes, "A statistical theory of learning," *Psychological Review, vol 57*, pp. 94–107, 1950.

[12] I. S. Mackenzie and nd R. W. Soukoreff, "Phrase sets for evaluating text entry techniques," in *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems CHI '03*, (Fort Lauderdale, Florida, United States), pp. 766–767, ACM, 2003.

[13] I. S. MacKenzie, H. Kober, D. Smith, T. Jones, and E. Skepner, "Letterwise: prefix-based disambiguation for mobile text input," in *Proceedings of the 14th annual ACM symposium on User interface software and technology*, UIST '01, (Orlando, Florida, United States), pp. 111–120, ACM, 2001.

[14] C. L. James and K. M. Reischel, "Text input for mobile devices: comparing model prediction to actual performance," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '01, (Seattle, Washington, United States), pp. 365–371, ACM, 2001.

[15] K. Kukich, "Techniques for automatically correcting words in text," *ACM Comput. Surv.*, vol. 24, pp. 377–439, December 1992.

 ISBN: 978-1-61208-236-3