

Chording with Spatial Mnemonics: Automatic Error Correction for Eyes-Free Text Entry

FRODE EIKA SANDNES AND YO-PING HUANG*

*Department of Computer Science
Oslo University College
0130 Oslo, Norway*

**Department of Computer Science and Engineering
Tatung University
Taipei, 104 Taiwan*

Chording is a technique that allows users to enter text without visual feedback. Traditional chording strategies are hampered by the substantial training effort required as users need to memorize chords. In this study an alternative chording scheme that uses spatial mnemonics to accelerate learning is proposed. Users mentally visualize the appearance of each character as a 3×3 pixel grid. The grid is input as a sequence of three chords. However, a problem with this approach is that the error probability accumulates across the three chords and the total error rate is therefore higher than for traditional chording. This study addresses to what degree these errors can be automatically corrected. Unlike traditional text correction approaches that operate with characters as the atomic unit, the redundancy available in the chords is used to improve the error correction. Experimental evaluations show that the strategy is capable of correcting 18.5% of all individual character construction errors. Next, the strategy is capable of correcting approximately twice as many word-level errors as MSWord, namely 69.5% of all substitution errors and 33.3% of all insertion errors. The strategy has at least two target audiences: users who need to devote their visual attention to other tasks and blind and visually impaired users who are unable to use visual intensive text entry techniques.

Keywords: mobile text entry, chording, error correction, miniature device, eyes-free operation, spatial mnemonics, visually impaired users

1. INTRODUCTION

This study addresses a specific sub-area of text entry, namely automatic correction of text that may be entered eyes-free on miniature and mobile devices. The study of text entry has gained popularity over recent years due to the widespread use of mobile phone messaging. Users wish to enter text easily and efficiently on miniature mobile devices. Further, the popularity of devices increases as the devices are getting smaller, since they consequently become more mobile and easy to transport. However, as devices are getting smaller there is less physical space for keyboards. If the size of the keys is reduced the text entry error rate will soar because of Fitts' law [1]. Consequently, fewer keys are commonly used. For instance, a typical mobile phone has a 12-key numeric keypad. Smaller devices such as wristwatches and MP3 players often have as few as two keys for text entry.

Received August 16, 2005; accepted January 17, 2006.

Communicated by Jhing-Fa Wang, Pau-Choo Chung and Mark Billinghurst.

Many of the techniques proposed over recent years are heavily dependent on visual feedback for successful operation [2-9], and they are consequently unsuitable for blind and visually impaired users [10] or users who need to allocate their visual attention to different tasks, for example walking [11] or operating machinery [12]. In this study the term eyes-free operation refers to the task of retrieving, or producing, the characters of the alphabet without visual feedback, and not the text composition task itself as users may visually validate the text that has been input. However, in eyes-free text composition tasks the users do not even visually validate the input text. Obviously, errors that are not identified cannot be corrected and one is therefore reliant on the restoration of the intended meaning, either manually by the reader or automatically by the text entry subsystem. Note that this study is limited to the input of alphanumeric text as used to transcribe most western languages including English. The input of more complex language systems such as Traditional Chinese is beyond the scope of this study.

1.1 Key-Based Text Entry

The QWERTY text input strategy is by far the most widespread and high text entry speeds can be achieved eyes-free. Users must memorize the keyboard layout in order to type touch. This can take a long time as there is no apparent logic for the layout and users often need more than 10 hours of practice before reaching reasonable typing speeds. However, QWERTY keyboards can also be used by novices as they can intuitively hunt and peck for keys.

Devices that are unable to accommodate full keyboards must provide access to the characters by other means. One common approach is many-to-one character-to-key mappings, where the number of keys is reduced at the expense of additional keystrokes. Numeric keypad text entry on mobile phones is a good example of this. Characters can be retrieved using the so-called multi-tap approach [19], where the user taps a key, labeled with the desired character, repeatedly until the desired character is displayed. I.e., sequences of multiple keystrokes must be entered to retrieve a character. Another strategy is the date-stamp approach where text is entered using only three keys by scrolling through a wheel-of-letters [16]. This approach can be found on old arcade game machines and databank wristwatches. Many other sequential keystroke techniques have been proposed in recent years, for instance adaptive three-key techniques [16], static three-key text entry techniques [30], three-key wristwatches [23], four-key text entry technique [8] and five- key strategies [3, 10, 25]. Sequential text entry strategies such as the ones mentioned need the users to establish a mental model of the text entry strategy and visual feedback is required to keep track of the current state. It is difficult to mentally track a large number of states without visual feedback. These techniques are thus in general heavily reliant on visual feedback. However, they are often fast and intuitive to learn, especially if they are organized in terms of existing interaction paradigms such as menu selection or item scrolling. Existing interaction skills can therefore be used to perform sub-tasks of the text entry task. Multi-keystroke text entry strategies are often compared in terms of their KeyStrokes Per Character (KSPC) [11, 15, 32] – a theoretical, and sometimes misleading, measure of effort. Various strategies for assessing errors have also been proposed [17, 32, 33].

1.2 Speeding up Text Entry with Language Models

Popular text entry optimizations include text entry prediction and disambiguation. Text prediction [6, 34], originally developed for disabled users, involves predicting the text the user is entering and completing words and sentences based on word and phrase prefixes. Dasher [35] is another variation on the theme where a language model is used to speed up text entry.

Disambiguation [19] is a technique especially used to enter text on mobile phones using the numeric keypad, but has also been applied to three-key input [29], four-key input [34] and five-key input [7, 25, 28]. Multiple characters are mapped to each key, one keystroke is used to represent each character and a wordlist is used to resolve the ambiguity from the sequence of keystrokes entered to compose a word. Often the word ambiguity cannot be resolved automatically and human intervention is needed to resolve the ambiguity. Both prediction and disambiguation are therefore heavily dependent on visual feedback as the user often has to choose between different alternatives, being it a set of predicted words, or a set of ambiguous words. These techniques are therefore not suitable for eyes-free operation

1.3 Chording

The study of chording has a long history (see the excellent survey by Noyes [21]). As Douglas Carl Engelbart invented the mouse in the 1960's he envisaged users holding the mouse in one hand and a piano-like chording keyboard in the other allowing users to execute pointing tasks and input text simultaneously [1].

The principle of chording is that users enter different chords with a limited number of keys to retrieve different symbols – much like the way musicians hits chords on the keys of a piano. Chording can be performed using one hand and one handed chording keyboards are often equipped with five keys for text entry, i.e., one key for each finger, although other configurations exist such as the Twiddler keyboard [14]. These keyboards can be used eyes-free as there is no need to look at the keyboard to determine where the fingers should be placed next as the fingers are not moved from one key to another. Several studies addressing various aspects of five-key chord keyboards exist [9, 24, 31]. Two handed chording keyboards are used to obtain higher text entry rates by overlapping the chords across hands [9]. Other examples of two handed chording keyboards are the stenograph [2] and Palantype that are used in courtrooms to transcribe text at speaking rates (approximately 180 words per minute).

Chording keyboards with five keys allow 31 unique chords to be entered and thus allow the English alphabet to be fully addressed. Although allowing for fast text entry rates, such systems have a high learning threshold because the users must remember the character chord patterns. These are usually simple binary patterns and it is hard to mentally make a connection between a chord and a character. This may be one of the reasons chording is not widely embraced, although Gopher and Raij [9] claim that chording in principle is easier to learn than QWERTY touch typing. The Microwriter is one commercial chording device that employs a set of different mnemonic aids to simplify the learning and recall of chords, including, spatial, word associations and kinaesthetic based. Its documentation claims that users can learn to use the Microwriter in 2.5 hours. Training time for chording devices has also been greatly reduced for users mastering the Braille

alphabet [4] – a reading system for the blind where characters usually are represented using 3×2 grids of elevated dots in various combinations. By using these Braille patterns directly as chording patterns on two handed chording devices with three keys for each hand, users are able to enter text with virtually no training as there is no need to learn the set of chords. Redundancies in the Braille alphabet can also be used to correct errors [27]. Unfortunately, few people are familiar with Braille.

Chording keyboards are either symmetric or asymmetric. Engelbart's piano-like keyboard is symmetric and can be used by either the left or the right hand, while the Microwriter has an asymmetric ergonomic design which can only be used by the right hand. Furthermore, chording skills can either be mirrored from one hand to the other or the spatial congruence of the chords are maintained, i.e., the leftmost finger on the right hand is still used as the leftmost finger on the left hand etc. The ingenious half QWERTY keyboard [20] is a good example of an approach that relies on the mirroring of typing skill. Besides its asymmetric design the Microwriter is also problematic from the viewpoint that it uses a mixture of different types of mnemonics, some of which are mirrored (kin-aesthetic, i.e., the agile index finger is used for the highly frequent letter E or word association, i.e., S for signet finger) and others that maintain spatial congruence (spatial mnemonics, i.e., the spatial layout of A) which means that it cannot easily be swapped between hands. The chording strategy studied herein is likely to maintain spatial congruence as it is based on spatial mnemonics. A consequence of this is that it can be used by either the left or the right hand by both users with left hand dominance and right hand dominance provided the ergonomic design of the keyboard is symmetric. However, hand-to-hand skill transfer is not addressed in this study.

1.4 Automatic Correction of Spelling Errors

There is a vast body of literature addressing automatic correction of text (see the excellent survey by Kukich [12]). Most approaches classify spelling errors into three categories, namely substitutions, insertions and deletions [5]. A substitution occurs when one character has been replaced with a different character, an insertion occurs when an additional character is inserted somewhere in a word and a deletion occurs when a character is missing from a word. A hybrid mixture of techniques is typically used to correct errors. One strategy is to maintain a list of common misspellings. Another approach is to systematically check other permutations of the characters in a word against a wordlist based on various levels and combinations of substitutions, insertions and deletions. Phonetic strategies such as SOUNDEX and Metaphone [22] are also good at finding viable alternatives to the incorrectly spelled words where words are matched according to their sound of pronunciation and not their spelling. Automatic correction of text is a difficult problem and state of the art tools, such as MSWord, are therefore often reliant on user intervention.

2. CHORDING WITH SPATIAL MNEMONICS

2.1 Input Strategy

Mnemonics are often used to accelerate learning, for example, learning the spatial

layout of a virtual keyboard [13]. In this study spatial mnemonics are used to assist the input of characters. A character is entered in four steps. First the user visualizes the character as a 3×3 grid of pixels. Then the user inputs this character grid in three steps from top to bottom. In each step one chord is entered representing one scan-line of the character grid. For each chord one, two or three keys are used simultaneously. After the three chords are entered the required character is retrieved and the process is repeated for all subsequent characters.

For example, imagine the user wishes to input the character ‘T’. The user first visualizes the character as a 3×3 grid of pixels (see Fig. 1). Then the user enters the three chords – the first scan-line requires the user to press all the three keys simultaneously (L + M + R) where L = left key, M = middle key and R = right key, and then release the keys. Next, the second and the third chords are simply the middle key (M). (Thus, the complete keystroke sequence required to enter ‘T’ is (L + M + R → M → M)).

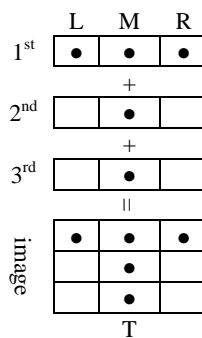


Fig. 1. 3×3 pixel grids (chord pattern) for the character ‘T’. L denotes left key down, M the middle key down and R the right key down.

2.2 Spatial Mnemonics and Cognitive Model

The mental visualization of the 3×3 grid is based on the assumptions that the users are familiar with the spatial appearance of the symbols they wish to input, and that they are able to mentally map these mental images onto the physical 3×3 pixel grids. It is likely that most users are able to infer mappings for simple characters such as ‘T’ and ‘L’ without training. Further, it is also likely that most users are able to map more complex symbols such as ‘M’ and ‘G’, which do not naturally fit onto a 3×3 pixel grid, with some training. Most users should be able to memorize these difficult patterns after seeing a few repeated examples.

The text entry strategy discussed herein can be realized using a simple cognitive model. No memorization of keyboard layouts or chord patterns is necessary since the user can infer the 3×3 pixel grids from their knowledge about the spatial structure of the character. This means that users can start entering text with very little training. A timeout-reset allows users to easily synchronize with the system after interruptions or pauses. The users learn that if they pause too long (one second) while entering the chords the user must re-enter the character from the beginning. The timeout mechanism is also use-

ful if users get confused about which state they are in – they can simply pause and try again. Simple errors can therefore be corrected manually without visual clues, since the time- delay can be determined, or “felt”, without visual feedback.

2.3 The Character Map

The chord patterns devised for this study are listed in Table 1. Their design is based on a mixture of technical considerations and the subjective opinions of several of the first author’s colleagues and students. Most characters are unproblematic and unambiguous to map, for example, ‘C’, ‘D’, ‘H’, ‘I’, ‘J’, ‘L’, ‘O’, ‘T’, ‘U’, ‘V’, ‘X’, and ‘Y’ although there are a few minor possible variations, such as whether the ‘C’ should be “curved” ($M + R \rightarrow L \rightarrow M + R$) or comprise three line segments ($L + M + R \rightarrow L \rightarrow L + M + R$). A stylistic version of the ‘I’ was used, i.e., two horizontal lines connected by a vertical line ($L + M + R \rightarrow M \rightarrow L + M + R$), instead of the obvious and simplistic vertical straight line ($M \rightarrow M \rightarrow M$), which instead was reserved for the more frequent SPACE character.

Table 1. Chord patterns for the characters used in this experiment.

A	B	C	D	E	F	G	H	I	J
###	#..	..##	##.	###	###	##.	##	###	..#
###	###	#..	##	#..	##	###	###	..#	..#
##	##	..##	##.	###	..	###	##	###	##.
K	L	M	N	O	P	Q	R	S	T
##	#..	..##	..##	..##	###	###	##.	#..	###
##	#..	###	##	##	###	##.	##	###	..#
##	###	###	##	##.#	##	..#	..#
U	V	W	X	Y	Z	space	.	enter	Back-space
##	##	..##	##	##	##.	..	###	..#	..
##	##	###	###	..#	..#
###	..	###	##	#..	..##	..	###	###	..#

Furthermore, with some imagination it is also straightforward to visualize mappings for ‘A’, ‘E’, ‘F’, ‘K’, ‘P’ and ‘R’. However, it is not obvious how to map the letters ‘S’ and ‘Z’, but by departing from the notion of a curved S and instead representing it by three connected right-angled line segments, and representing ‘Z’ by the mirror image of ‘S’, the resulting patterns become simple and identifiable representations of the respective characters. Next, it was found that it was easier to represent ‘B’ and ‘N’ using their lowercase representation instead of their uppercase representation, although this results in a mixture of uppercase-lowercase mnemonics. The remaining characters ‘G’, ‘M’, ‘Q’ and ‘W’ are graphically more complex and their realization are provided in Table 1.

2.4 Character-Level Error Recovery

Error correction is achieved exploiting redundancy in information [26]. Simple chord-level error-recovery can be performed on the chords if redundancy is incorporated

into the character sets. To the best of our knowledge there are no other keyboard based text entry strategies that can perform error-recovery below word level. Existing strategies employ error recovery at word level using dictionaries.

The 3×3 pixel grid can be viewed as a vector of 9 bits which can represent 512 different patterns. However, it is not realistic to exploit the entire capacity for standard text entry. If we wish to represent the letters of the English alphabet (26), numbers (10), navigation, editing and case (11) and punctuation and other symbols (33) then this adds up to 89 symbols, which yield 82% redundancy. This redundancy can be exploited to correct errors. If the user makes a mistake, while entering one of the chords, the closest pixel grid in terms of Hamming distance is selected. Fig. 2 shows an example of the last chord of 'L' being incorrectly input as (L + M) instead of (L + M + R), but the error is detected, corrected and the character 'L' is output. Clearly, there are several situations where errors lead to ambiguities and the proposed strategy is unable to resolve these ambiguities. See for example Fig. 3 which shows the pixel grids for the characters 'A' and 'O' and an unknown erroneous pixel grid with a Hamming distance of 1 to both 'A' and 'O'.

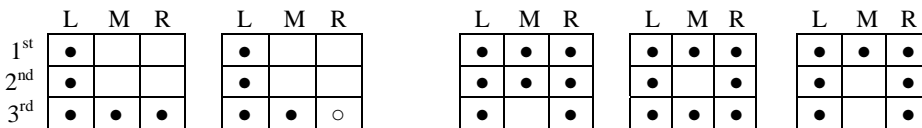


Fig. 2. The chords for the characters 'L' and an erroneously entered 'L' which can be automatically corrected.

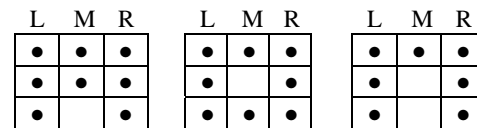


Fig. 3. The chords for the characters 'A', 'O' and an unknown erroneously entered pixel grid resulting in error-correcting ambiguity.

The strategy in its general form is capable of successfully correcting some single bit errors and a few double bit errors. A single bit error means that too many or too few keys in one of the three chords have been pressed accidentally.

Note that character level error correction takes place once a character is completed and the user can observe the effect on the display immediately. Character level error correction is therefore particularly useful for textual interaction with visual feedback.

2.5 Word-Level Error Recovery

Errors that cannot be corrected at character level, as described in the previous section, may be resolved at word level by the means of a language model. Once a word is completed it can be matched against a wordlist as is commonly done by classic spell-checking software. However, traditional spell-checkers operate with characters as the atomic unit of information. In contrast, the sequence of text input chords provides more information about the desired word as the state of each finger in a chord is the atomic unit of information. Therefore, a chord may be partially correct if the user accidentally entered one of the chords incorrectly, or is unable to completely memorize the exact spatial representation of a character. This partial information can be used to up the error-correction rate by finding better and more exact matches in the wordlist. The following word representation strategy is used for both the input word and the wordlist. Each chord

comprises a 3-bit vector, and a character is therefore a 9-bit vector. A word is consequently a $9n$ -bit- vector, where n is the length of the word. First, one checks if the input word exists in the wordlist. If it does not it is considered incorrect. If one or more of the chords are incorrectly entered then there are one or more substitution errors. To correct these errors the bit representation of the word is matched against all other words in the wordlist with matching lengths and the one with the smallest Hamming distance is selected.

One drawback of using the Hamming distance as a measure of similarity is that all the bit errors have equal weights. The implication is that a word in the wordlist differing by one bit in two different character positions has the same similarity as a word with two bits differing in one character position. Intuitively, one would expect the latter to be a better match. To overcome this problem one can use a modified Hamming distance measure of similarity where the Hamming distance is multiplied by the number of non-matching characters in the word. In the first example the score would be four, while it would still be two for the second case.

Insertion errors occur when users accidentally insert additional characters. To correct single insertion errors each character of the word is discarded in turn and the resulting word is checked against the wordlist. For example, the word "TOG" with the accidentally inserted 'G' is transformed into "OG", "TG" and the valid word "TO".

Note that word-level error correction only can take place once a word has been completed and the incorrectly entered characters can therefore not be updated on the display before the word is completed. It is therefore less attractive for visual intensive interactions as users may be tempted to make corrections and consequently interrupt the text composition task. However, it is totally unproblematic for eyes-free interaction.

3. EXPERIMENTAL EVALUATION

3.1 Overview

The text entry experiment was designed to obtain a profile of the errors users make while entering text using the strategy described herein. The users were asked to correct errors as they occurred (self reported errors) and only character level error correction was employed during the text entry experiments. To test the feasibility of using a language model, the acquired data was retrofitted into the proposed word-level error-correction model.

3.2 Subjects

A total of 14 subjects participated in the experiment. Subjects were recruited from the student population of Oslo University College, using e-mail advertisements, and it included 2nd year, 3rd year and master-level computer science students. All the subjects were male and ranged in age from 21 to 38 years with a mean age of 26 years. All subjects reported using a mobile phone on a daily basis with the exception of one subject reporting using a mobile phone occasionally. Nine subjects admitted sending SMS-messages on a daily basis, two on a weekly basis and three occasionally. Three of the

subjects reported being fast typists on mobile phones, nine subjects reported being ok or average and two subjects reported being slow typists on mobile phones. On average the subjects assessed that they spend 49 hours a week using a computer. Three of the subjects reported being very fast typists on a regular keyboard and the other subjects reported being average or ok. Regarding computer gaming habits, three subjects admitted playing computer games regularly, nine occasionally, and two subjects never play computer games. All the subjects reported being right-handed, having no disabilities including visual impairment and none reported being dyslexic. None of the subjects had experience entering text eyes free. On the attitude to new technology twelve subjects reported being very interested in new technology and the remaining two subjects were moderately interested in new technology. Each subject was paid 100 Norwegian kroner (approximately 10 Euros) for each hour of participation in the experiment, and a bonus of 100 kroner was awarded upon completing all the text entry sessions.

3.3 Experimental Setup

Standard desktop computers with monitors and full QWERTY keyboards were used to ensure a repeatable and controlled experiment. The software was custom implemented as two Java Applets running in a web browser. One applet comprised the character pattern reference and was presented as a virtual QWERTY keyboard. By clicking on one of its keys a graphical representation of the corresponding 3×3 pixel map for the character was displayed for the duration of 1 second. The time-restriction was devised to more easily record the effect of practice on learning. The second applet comprised the text entry application. The applet had two components. The top line showed a phrase to be copied and at the bottom a multi-line text entry area was provided. As users entered text the text was displayed in the text entry area which behaved in the same manner as a conventional text-area control, much like the text-area often found in HTML forms. No intermediate state information was shown on the screen during the assembly of a character – this allowed the claim of eyes-free character input to be demonstrated. The text scrolls automatically so that it is possible to see a trail of previously entered phrases. Cursor navigation and editing was implemented, but the subjects were not instructed in these. The right side of the screen was used to display short instructions on how to operate the system. Both applets logged user activity locally to disk.

The applet was configured to accept keystrokes from the SPACE, J and K keys as this is a hand position touch typists are used to that naturally fits the thumb, index finger and long finger of the right hand. This is an asymmetric finger configuration that may introduce some stimulus-response incompatibility as the SPACE-bar is not aligned on the same line as J and K. I.e., the finger positions are two-dimensional while the chord scanline is one-dimensional. However, experimentation revealed this to be more comfortable than using the three keys in a row such as H, J and K. Devices such as mobile phones could not be used in this experiment as the physical characteristics of their numerical keypad makes them unsuitable for chording tasks.

3.4 Procedure

The experiment was carried out in isolation. Each subject was instructed in how to

operate the text entry system and the text entry task to be conducted. The principal investigator was present during the text entry sessions to answer questions and provide help. Each session lasted one hour and a total of four sessions were carried out for each of the 14 subjects. Each session was separated by approximately three days.

3.5 Materials

Four different lists of phrases randomly selected from the phrase sets described in [18], with a mean phrase length of 5.4 words, were used in the experiments. Each phrase list was used for each of the four sessions to eliminate phrase learning effects. All phrases were presented in lowercase as lowercase characters are easier to read, and the phrases did not contain special characters or punctuation symbols. The users had to perform an implicit mental transformation from lower to upper case as the text was mostly displayed as lower case characters, while each character is input as an uppercase pattern.

3.6 Task

The users were asked to conduct a text copying task where the phrases were displayed one by one. Once completing a phrase the user had to enter the chord sequences for ENTER to display the next phrase. There was no facility to go back. Subjects were instructed to correct errors (self reported errors) using the chord sequences representing BACKSPACE. The copying task limits the range of errors that can occur as users always see how a word should be spelled.

3.7 Measurements

The character reference applet logged the character looked up by the subjects and the time at which the reference was made. The text entry applet logged all the keystrokes with associated timestamps and the input phrases and text output by the system.

3.8 Analysis

A number of scripts were written to extract the relevant information from the log files which are presented in this study. The various parameters discussed herein are based on means for each subject over an entire session.

Statistical significance tests were performed using both a one-way ANOVA and two-way repeated measures ANOVA using the Microsoft Excel analysis tool pack. A significance level of 0.05 was used. The session was the independent variable, the subjects were used as replications (no between subject analysis was performed) and the dependent variables include: error-rates and text entry speeds. The error-correction strategy used was the second factor (independent variable).

3.9 Retrofitting Text Entry Behavior

The self reported errors were used to establish a set of uncorrected-corrected word pairs, namely the word as it would appear without any corrections and the correct word

obtained after all the manual corrections are applied. The character input sequence was first tokenized with SPACE and ENTER as separators. Next, the uncorrected and corrected words were extracted by removing or applying all corrections from the input sequence respectively. The pattern for a correction is A + BACKSPACE + B, where the user first enters character A, inputs BACKSPACE to delete the A, and then enters the character B. In this instance the uncorrected sequence is A, and the corrected sequence is B. This extends to longer sequences as well. For instance, if the user enters A + BACKSPACE + B + BACKSPACE + C then clearly the uncorrected word is A and the corrected word is C. Such sequences indicate trial-and-error behavior.

Another common correction pattern has the following characteristics: A + B + BACKSPACE + BACKSPACE + C + B. In this situation the user inputs multiple characters AB, discovers that the first character in the sequence (A) is incorrect, goes back to that character by inputting BACKSPACE twice, then enters the desired character C followed by B. Clearly, in this instance the uncorrected sequence is AB and the corrected sequence is CB. This correction pattern can be termed go-back-N, where N denotes the number of consecutive BACKSPACE characters in the sequence.

Chord based error correction was applied to the list of uncorrected, corrected word pairs of each subject, and a success was recorded once the automatically corrected version of the uncorrected word matched the correct version of the word. The bit representation of each uncorrected word was matched against the entries of an English wordlist with 80,736 entries taken from a standard Debian Linux distribution and the most similar word was selected. For comparison the uncorrected words were also corrected using Microsoft Word, which can be viewed as state of the art in classic character level spell-checking and correcting. MSWord automatically identifies incorrectly spelled words and presents a ranked list of viable alternatives where the most probable alternative is highest on the list. In order to conduct a repeatable experiment the default alternative was consistently selected.

4. RESULTS

4.1 Error Profile

The mean character input error rate for all the four sessions is 10.8% with a large spread ($SD = 7.4$) and there was no significant effect on practice ($F(1, 52) = 0.41$; $p > 0.75$). This is high as most literature on text entry report error rates of about 5%.

Fig. 4 shows the distribution of error types observed during the text entry experiment in terms of word-level errors. Clearly, a majority (19.0%) of the errors are substitution errors. Next, 8.1% of the errors are insertions and other errors including deletions account for only 0.9%. The difference between the different error types is highly statistically significant ($F(2, 156) = 118.1$; $p < 0.001$), but there is no significant difference across the sessions ($F(3, 156) = 0.13$; $p > 0.94$). This distribution is likely to be influenced by the text copying task where there is no confusion regarding the spelling of words. The distribution is likely to be different for text composition tasks where the users must rely on their own knowledge of spelling.

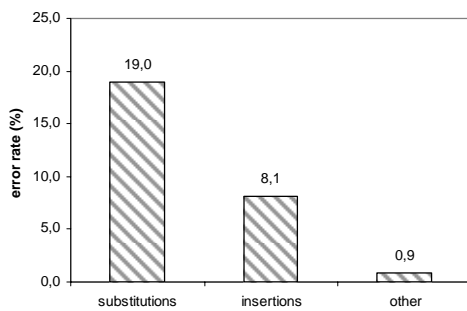


Fig. 4. Distribution of errors types in terms of word-level errors.

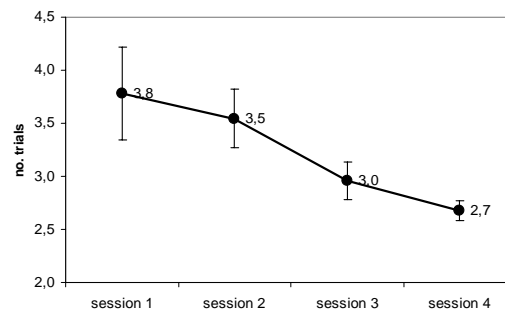


Fig. 5. Mean number of trial-and-errors per character. Error-bars show standard deviation.

The high portion of substitution errors suggests user trial-and-error behavior, i.e., users approximately can recall the appearance of the character, but not exactly and therefore have to make multiple attempts before getting it right. Fig. 5 shows the mean number of attempts needed to retrieve the desired characters. During the practice session a mean of 3.8 attempts were made before the desired character was retrieved and this mean falls to 2.7 attempts during the final session. Therefore, practice helps reducing the number of attempts ($F(1, 52) = 48.8; p < 0.001$).

4.2 Character-Level Correction

The character level error corrector successfully corrected 2.0% of all the inputted characters, and the effective text entry rate was therefore 8.8%. This means that 18.5% of all character construction errors were automatically corrected based on the redundancy available in the chords. The character-level error correction is applicable if the user is visually inspecting the text as it is entered. However, when text is entered eyes free it is better to apply word-level error correction directly since users are not relying on the immediate visual feedback. Word-level correction is likely to yield an overall higher correction rate as there is more redundancy in the language than in individual chords.

4.3 Word-Level Correction

Figs. 6 and 7 show the ratio of automatically corrected substitution and insertion errors using the method described herein with the improved Hamming distance. Similar results obtained using MSWord is provided for reference. Fig. 6 shows that the proposed error correction strategy is capable of correcting 69.5% of all words with substitution errors. This is significantly better than what is achieved using MSWord which is capable of correcting only 30.2% of all the substitution errors ($F(1, 104) = 355.1; p < 0.001$). Furthermore, Fig. 7 shows that the proposed strategy is capable of correcting 33.3% of all the insertion errors, while in comparison MSWord is only capable of correcting 16.1% of the insertion errors, which is significantly less ($F(1, 104) = 27.3; p < 0.001$). The results demonstrate that the simple strategy for correcting text entry errors based on the Hamming distance between the spatial chord-representation of characters is far more effective in correcting substitution and insertion errors than the more complex spelling corrector

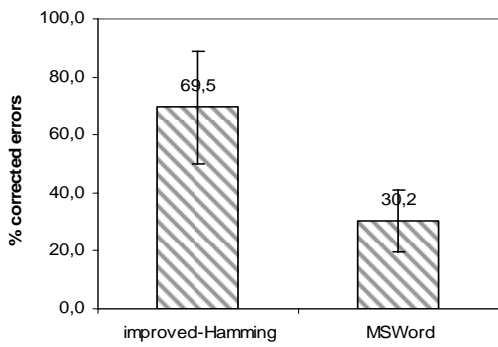


Fig. 6. Ratio of automatically corrected substitution errors using the proposed strategy and MSWord. Error-bars show standard deviation.

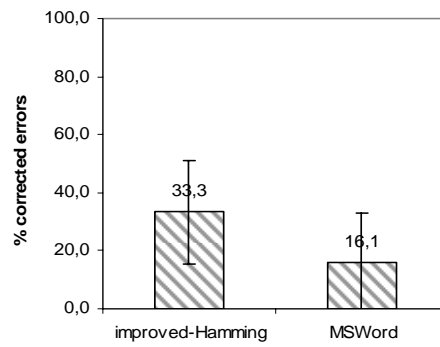


Fig. 7. Ratio of automatically corrected insertion errors using the proposed strategy and MSWord. Error-bars show standard deviation.

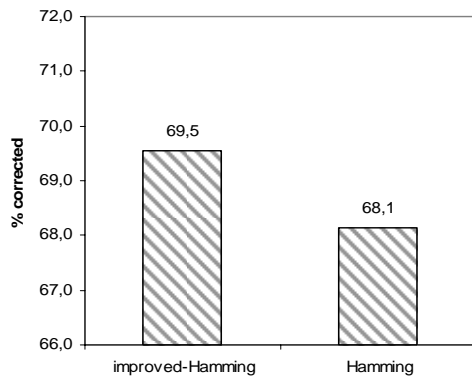


Fig. 8. Correcting substitution errors with Hamming based and improved Hamming based measure of similarity.

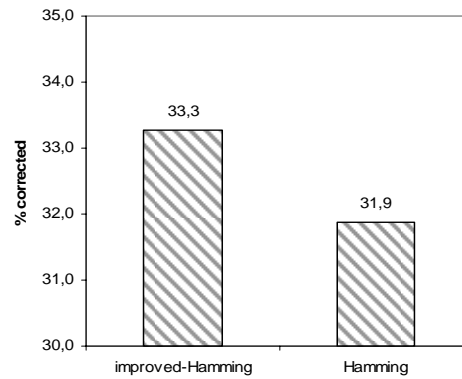


Fig. 9. Correcting insertion errors with Hamming based and improved Hamming based measure of similarity.

available in MSWord which operates with characters as the atomic unit of information. It is therefore highly likely that the error correction rate could be further improved by combining some of the classic linguistics based spelling correction heuristics with the partial information available in the chord patterns.

Figs. 8 and 9 shows the difference of using a Hamming based measure of similarity and an improved Hamming based measure of similarity where the number of characters that differs in the two words is used to scale the Hamming distance. In both instances the improved Hamming distance is marginally better than the plain Hamming distance. The improvements are consistent across the dataset although the differences are not statistically significant due to the large spread in the data ($F(1, 104) = 0.30; p > 0.57$ and $F(1, 104) = 0.16; p > 0.68$, respectively).

Since substitution errors are the easiest to correct and they occur with the highest rate one can roughly estimate that given similar error profiles as the one observed in this

study the overall error rate can be reduced from 28.0% erroneously entered words to 12.5% erroneous words.

4.4 Implications for Text Entry Performance

Fig. 10 shows the actual (uncorrected) text entry performance achieved over the four session measured in words per minute (WPM). This measure is based on correctly entered text as erroneous input is discarded. Furthermore, the time delay that occurs between displaying a new phrase to be copied and the first keystroke is also discarded to minimize reading effects.

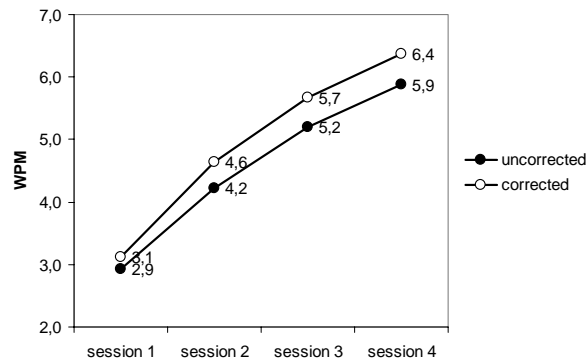


Fig. 10. Potential performance improvement as a result of automatic error correction.

Clearly, there is an obvious effect of practice as users get accustomed to the text input technique ($F(3, 48) = 6.06; p < 0.001$). The plot also shows an estimate of the achievable performance if the users do not make manual corrections while entering text, and is particularly applicable to users entering text eyes-free. The estimate is computed by discarding the accumulated time associated with all the correctional keystrokes. During the practice session there would be a slight increase from 2.9 WPM to 3.1 WPM, while after four hours of practice there is a potential increase from 5.9 WPM to 6.4 WPM. This is nearly a 10% increase in text entry performance due to the elimination of the unnecessary correctional keystrokes.

5. CONCLUSIONS

Spatial mnemonics can be used to accelerate the learning of a chording based text entry for non-expert users, and text can hence be entered without visual feedback. The results of this study show that although the technique yields more errors than other strategies it is possible to compensate for this by successfully correct a majority of these errors automatically. About 18.5% of all incorrectly constructed characters were automatically corrected interactively during the text entry experiment by matching them to the closest shape in the valid character map. Furthermore, by retrofitting the input data onto a word-

level correction model 69.5% of all substitution errors and 33.3% of all insertion errors were corrected using a simple strategy comprising of a wordlist and a similarity measure based on the spatial relationship of characters. The results using the bits of the chord as the atomic unit of information greatly outperformed the results achieved using MSWord which operates with the character as the atomic unit of information. Character level error correction is applicable to situations where text is entered with visual feedback and the word-level error correction strategy is particularly suitable for eyes-free text entry. Eyes-free text entry is not likely to result in trial-and-error behavior, and the elimination of trial-and-error keystroke sequences will result in higher productivity.

REFERENCES

1. S. B. Barnes, "Douglas Carl Engelbart: developing the underlying concepts for contemporary computing," *IEEE Annals of the History of Computing*, Vol. 19, 1997, pp. 16-26.
2. M. P. Beddoes and Z. Hu, "A chord stenograph keyboard: a possible solution to the learning problem in stenography," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 24, 1994, pp. 953-960.
3. T. Bellman and I. S. MacKenzie, "A probabilistic character layout strategy for mobile text entry," in *Proceedings of Graphics Interface*, 1998, pp. 168-176.
4. M. C. Cho, K. H. Park, S. H. Hong, J. W. Jeon, I. S. Lee, H. Choi, and H. G. Choi, "A pair of braille-based chord gloves," in *Proceedings of the 6th International Symposium on Wearable Computers*, 2002, pp. 153-156.
5. F. J. Damerau, "A technique for computer detection and correction of spelling errors," *Communications of the ACM*, Vol. 7, 1994, pp. 171-176.
6. J. J. Darragh, I. H. Witten, and M. L. James, "The reactive keyboard: a predictive typing aid," *IEEE Computer*, Vol. 23, 1990, pp. 41-49.
7. M. Dunlop, "Watch-top text-entry: Can phone-style predictive text-entry work with only 5 buttons?" in *Proceedings of the 6th International Symposium on Mobile Human-Computer Interaction*, LNCS 3160, 2004, pp. 342-346.
8. T. Evreinova, G. Evreino, and R. Raisamo, "Four-key text entry for physically challenged people," in *Proceedings of the 8th ERCIM Workshop on User Interfaces for All*, 2004.
9. E. Gopher and D. Raij, "Typing with a two-hand chord keyboard: Will the qwerty become obsolete?" *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 18, 1985, pp. 601-609.
10. P. Isokoski and R. Raisamo, "Device independent text input: a rationale and an example," in *Proceedings of the Working Conference on Advanced Visual Interfaces*, 2000, pp. 76-83.
11. 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*, 2001, pp. 365-371.
12. K. Kukich, "Techniques for automatically correcting words in text," *ACM Computing Surveys*, Vol. 24, 1992, pp. 377-437.
13. P. U. J. Lee and S. Zhai, "Top-down learning strategies: can they facilitate stylus

- keyboard learning?" *International Journal of Human Computer Studies*, Vol. 60, 2004, pp. 585-598.
14. 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*, 2004, pp. 671-678.
 15. I. S. MacKenzie, "KSPC (keystrokes per character) as a characteristic of text entry techniques," in *Proceedings of the 4th International Symposium on Human-Computer Interaction with Mobile Devices*, LNCS 2411, 2002, pp. 195-210.
 16. I. S. MacKenzie, "Mobile Text entry using three keys," in *Proceedings of the 2nd Nordic Conference on Human-Computer Interaction*, 2002, pp. 27-34.
 17. I. S. MacKenzie and R. W. Soukoreff, "A character-level error analysis technique for evaluating text entry methods," in *Proceedings of the 2nd Nordic Conference on Human-Computer Interaction*, 2002, pp. 243-246.
 18. I. S. MacKenzie and R. W. Soukoreff, "Phrase sets for evaluating text entry techniques," in *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems*, 2003, pp. 754-755.
 19. I. S. MacKenzie and R. W. Soukoreff, "Text entry for mobile computing: models and methods, theory and practice," *Human Computer Interaction*, Vol. 17, 2002, pp. 147-198.
 20. A. E. Matias, I. S. MacKenzie, and W. Buxton, "Half-QWERTY: typing with one hand using your two-handed skills," in *Companion of the Conference on Human Factors in Computing Systems*, 1994, pp. 51-52.
 21. J. Noyes, "Chord keyboards," *Applied Ergonomics*, Vol. 14, 1983, pp. 55-69.
 22. L. Phillips, "Hanging on the metaphone," *Computer Language*, Vol. 7, 1990, pp. 39.
 23. M. T. Raghunath and C. Narayanaswami, "User interfaces for applications on a wrist watch," *Personal and Ubiquitous Computing*, Vol. 6, 2002, pp. 17-30.
 24. R. Rosenberg and M. Slater, "The chording glove: a glove-based text input device," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 29, 1999, pp. 186-191.
 25. F. E. Sandnes, "One handed text entry: evaluation of five-key text entry techniques," in *Proceedings of the IFIP TC8 Working Conference on Mobile Information Systems*, 2004, pp. 331-339.
 26. F. E. Sandnes, "Portable multitrack audio storage strategy based on Reed Solomon codes," *IEE Electronics Letters*, Vol. 35, 1999, pp. 631-632.
 27. F. E. Sandnes and Y. P. Huang, "Chord level error correction for portable Braille devices," *IEE Electronics Letters*, Vol. 42, 2006, pp. 82-83.
 28. F. E. Sandnes and H. L. Jian, "Pair-wise variability index: evaluating the cognitive difficulty of using of mobile text entry systems," in *Proceedings of the 6th International Symposium on Mobile Human-Computer Interaction*, LNCS 3160, 2004, pp. 347-350.
 29. F. E. Sandnes, H. W. Thorkildssen, A. Arvei, and J. O. Boverud, "Techniques for fast and easy mobile text-entry with three-keys (dictionary based)," in *Proceedings of NIK2003 – Annual National Norwegian Computer Science Conference*, 2003, pp. 205-216.
 30. F. E. Sandnes, H. W. Thorkildssen, A. Arvei, and J. O. Boverud, "Techniques for fast and easy mobile text-entry with three-keys (non-dictionary based)," in *Proceed-*

- ings of the Hawaiian International Conference on System Science, 2004.
31. R. Seibel, "Performance on a five-finger chord keyboard," *Journal of Applied Psychology*, Vol. 46, 1962, pp. 165-169.
 32. R. W. Soukoreff and I. S. MacKenzie, "Metrics for text entry research: a evaluation of MSD and KSPC, and a new unified error metric," in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 2003, pp. 113-120.
 33. R. W. Soukoreff and I. S. MacKenzie, "Recent developments in text-entry error rate measurement," in *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems*, 2004, pp. 1425-1428.
 34. K. Tamaka-Ishii, Y. Inutsuka, and M. Takeichi, "Entering text with a four-button device," in *Proceedings of the 19th International Conference on Computational Linguistics*, 2002, pp. 988-994.
 35. D. J. Ward, A. F. Blackwell, and D. J. C. MacKay, "Dasher – a data entry interface using continuous gestures and language models," in *Proceedings of the ACM Symposium on User Interface Software and Technology*, 2000, pp. 129-137.



Frode Eika Sandnes (杉福羅) received a B.Sc. in Computing Science from the University of Newcastle Upon Tyne, England, in 1993, and a Ph.D. in Computer Science from the University of Reading, England, in 1997. He has several years of experience from the space industry developing communications and onboard systems for low-earth orbit environmental satellites. He is currently an Associate Professor in the Department of Computer Science at Oslo University College, Norway, and he was a visiting professor at National Cheng Kung University, Tainan, Taiwan and Tatung University, Taipei, Taiwan during the spring of 2005. Dr. Sandnes' research interests include multiprocessor scheduling, error-correction and mobile human computer interaction.



Yo-Ping Huang (黃有評) received a Ph.D. in Electrical Engineering from Texas Tech University, U.S.A., in 1992. Dr Huang is currently a professor in the Department of Computer Science and Engineering at Tatung University, Taiwan. Professor Huang is also the Dean of research and development office and director of valuable instrument center at Tatung University. His research interests include intelligent information retrieval, data mining, artificial intelligence and application systems for hand-held devices.